MASTER'S THESIS | LUND UNIVERSITY 2015

Fast GPU Accelerated Stereo Correspondence

Sebastian Hindefelt, Vilhelm Sturén

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2015-33

Fast GPU Accelerated Stereo Correspondence

(for embedded surveillance camera systems)

Sebastian Hindefelt Vilhelm Sturén

May 18, 2015

Master's thesis work carried out at Axis Communications AB.

Supervisors: Michael Doggett
Yuan Song

Examiner: Per Andersson, per.andersson@cs.lth.se

Abstract

Many surveillance applications could benefit from the use of stereo cameras for depth perception. While state-of-the-art methods provide high quality scene depth information, many of the methods are very time consuming and not suitable for real-time usage in limited embedded systems. This study was conducted to examine stereo correlation methods to find a suitable algorithm for real-time or near real-time depth perception through disparity maps in a stereo video surveillance camera with an embedded GPU. Moreover, novel refinements and alternations was investigated to further improve performance and quality. Quality tests were conducted in Octave while GPU suitability and performance tests were done in C++ with the OpenGL ES 2.0 library. The result is a local stereo correlation method using Normalized Cross Correlation together with sparse support windows and a suggested improvement for pixel-wise matching confidence. Applying sparse support windows increased frame rate by 35% with minimal quality penalty as compared to using full support windows.

Contents

1 Introduction **7**
- 1.1 Background . . . 7
- 1.2 Aim and Purpose . . . 8
- 1.3 Method . . . 8
- 1.4 Disposition . . . 8
- 1.5 Division of Work . . . 9
- 1.6 Limitations . . . 9
- 1.7 Acknowledgments . . . 10

2 Technology Background **11**
- 2.1 History . . . 11
- 2.2 Related Work . . . 12
- 2.3 GPU . . . 13

3 Stereo Correspondence **17**
- 3.1 General Stereo problem . . . 17
  - 3.1.1 Epipolar geometry . . . 17
  - 3.1.2 Error analysis . . . 18
- 3.2 Global methods . . . 19
- 3.3 Semi-Global methods . . . 19
- 3.4 Local Methods . . . 20
  - 3.4.1 SAD . . . 21
  - 3.4.2 SSD . . . 21
  - 3.4.3 NCC . . . 21
  - 3.4.4 Zero-Mean Methods . . . 22
  - 3.4.5 Locally-Scaled Methods . . . 22
  - 3.4.6 Census Transform (SHD) . . . 23
  - 3.4.7 Hierarchical Block Matching . . . 23
  - 3.4.8 Evaluation of Cost Functions . . . 24
  - 3.4.9 Result and Analysis . . . 25

4 Refinements **27**
4.1 Support window size . . . 27
4.2 Confidence Metrics . . . 27
4.2.1 Uniqueness of minimum/maximum . . . 27
4.2.2 LR-RL consistency check . . . 28
4.3 Filters . . . 28
4.3.1 Median filtering . . . 29
4.3.2 Bilateral filtering . . . 29
4.3.3 Mathematical morphology . . . 29
4.4 Gap filling . . . 30
4.5 Results and Analysis . . . 31
4.5.1 Support window size . . . 31
4.5.2 Uniqueness of minimum/maximum . . . 32
4.5.3 LR-RL consistency check . . . 32
4.5.4 Filters . . . 34
4.5.5 Gap filling . . . 34

5 Investigations **35**
5.1 Disparity Distribution . . . 35
5.2 Number of Active Pixels in Support Window . . . 35
5.3 Patterns . . . 36
5.4 Every Other Disparity Match with Neighbor Control . . . 37
5.5 Results and Analysis . . . 37
5.5.1 Disparity Distribution . . . 37
5.5.2 Number of Active Pixels in Support Window . . . 39
5.5.3 Patterns . . . 39
5.5.4 Every Other Disparity Match with Neighbor Control . . . 41

6 Implementation **43**
6.1 Platform . . . 43
6.1.1 OpenCV . . . 43
6.1.2 OpenGL . . . 44
6.2 Implementation of Stereo Matching . . . 45
6.3 Optimizations . . . 45
6.3.1 RGBA used as YYYY . . . 45
6.4 Video Surveillance Applications . . . 48
6.4.1 Flexible speed measurements . . . 48
6.4.2 Depth mask . . . 48
6.4.3 Object size estimation . . . 48

7 Discussion and Conclusion **49**
7.1 Discussion . . . 49
7.1.1 Comparison to Related Work . . . 50
7.2 Conclusions . . . 50
7.3 Future Work . . . 51

8 Appendix **53**

Bibliography **61**

Abbreviations

AR Augmented Reality

FoV Field of View

GPU Graphics Processing Unit

GPGPU General Purpose Graphics Processing Unit

SIMD Single-Instruction Multiple-Data

SGM Semi-Global Matching

MI Mutual Information

FPGA Field-Programmable Gate Array

RMSE Root-Mean-Square Error

BMP Bad Matching Pixels

WCBiMF Weighted Cross-Bilateral Median Filter

Chapter 1
Introduction

1.1 Background

This study was carried out at Axis Communications AB which is a manufacturer of network video cameras for the surveillance industry. An embedded GPU will be included in the next generation of their network cameras. The unique setup will expand the possibilities for adding new exciting features directly in the cameras such as dynamic overlays and advanced video transforms. Many of their initial ideas for utilizing the GPU involve different types of Augmented Reality (AR) applications, but to be able to implement most of these it is crucial to have depth perception of the scene. The topic for this thesis was therefore a fundamental subject of all of these AR applications; disparity maps from stereo cameras.

For example, an important part of video surveillance is motion detection and object tracking. Without depth perception it is very difficult to determine size and identifying type of an object causing motion in the video stream. Motion detection often triggers due to other movements in the scene than human activities which results in false alarms. These alarms are expensive for the customers and could also be dangerous since frequent alarms may result in lowered attention for each alarm. Therefore it is of high interest for all involved parts to minimize the number of false alarms. Examples of common false alarm triggers are shadows, reflections in puddles on the ground and small animals in front of the camera which result in motions. Information about the depth in the scene could help detect and filter out some of these false triggers.

Depth perception of the scene will also allow for AR applications to be embedded in surveillance camera systems. A possible application of AR in video surveillance systems is to augment the objects that are interesting and possibly filter out objects that are perturbing the view or distracting the operator. This will make it easier for the user to extract information from images that is crucial for the specific situation and otherwise would be hard to distinguish. It will also be possible to place virtual objects in the scene.

1.2 Aim and Purpose

The objective of this thesis is to study existing methods for disparity map calculation and find the ones that are well suited for limited embedded GPUs. Also, by possibly allowing a quality trade off, find novel ways to reach real-time, or near real-time, performance on such limited hardware. The thesis will also aim to develop a flexible algorithm in which it is possible to alter the disparity map performance and quality depending on the requirements of the current application.

While some methods may produce disparity maps with very low errors it is nearly impossible to produce perfect results, therefore one aim is to be able to mark bad quality areas in the disparity map to aid post processing of the disparity map or applications using the disparity map.

The development of AR applications in the mobile industry is fairly well known today, but have yet to make any real impact on the surveillance industry. In addition to the development of an algorithm, the purpose of this thesis is to widen the field of application and give inspiration for further usage of GPUs in limited embedded systems, especially in the video surveillance field.

1.3 Method

To find a suitable method for calculating disparity maps a study of current methods were conducted. The result of the study was then evaluated for quality, performance and GPU implementation suitability. Further studies covered novel refinements and confidence measurements for improving the quality of the disparity map.

Prototyping and quality evaluations were done in Octave 3.6.2 using the *image* package. Performance was measured in C/C++ with the graphics library OpenGL ES 2.0 and GLSL 1.0 shader implementations. To fetch live images from the camera sensors and simulate the camera pipeline features such as stereo calibration and rectification, cropping, scaling and converting colors, a back end was implemented in C++ with OpenCV 2.4.

As comparison of measurements between methods the root mean square error (RMSE) and bad matching pixels (BMP) errors has been used.

1.4 Disposition

Initially the report introduce the basic concepts of stereo vision and disparity maps. A general explanation of GPUs and why it is suitable to accelerate disparity map computations using GPUs follows. Further the mathematics of disparity maps and epipolar geometry theory are presented and different methods for computing disparity maps are explained.

The thesis consists of three main parts: selection of a relevant method for computing disparity maps in this context, selection of suitable methods for improving the resulting disparity map and novel research towards increased performance and quality. These parts are addressed in chapter 3, 4 and 5 respectively. Each one of these three chapters are

divided into two parts: presentation of the concept or idea and their respective result and analysis.

Details about the implementation and information about the libraries used is found in chapter 6. The layout and data flow of the final algorithm is explained and possible uses for disparity maps with video surveillance in mind is presented in the same chapter.

Lastly, in chapter 7 discussions about the results and conclusions about the investigations and refinements are presented. Additionally, a comparison to related work is done where the methods and results of this study are compared to previous studies in the area. Suggestions for further work based on the result are also presented in this chapter.

1.5 Division of Work

Throughout this thesis the research, implementation and report writing has been divided equally between the students. Both students have devoted approximately the same amount of time over the course of the thesis work.

During the early phase Vilhelm was responsible for the research of related work, while Sebastian gathered the necessary theoretical material. Vilhelm covered local methods and Sebastian covered global and semi-global methods during the evaluation of stereo correspondence methods. In the refinement process Sebastian covered filtering and gap filling while Vilhelm evaluated support window size and disparity confidence metrics.

Vilhelm was responsible for Octave implementations and tests, while Sebastian developed the back end in OpenCV and coupled it with the OpenGL client during the investigation phase.

The main implementations in OpenGL were equally split and implemented by both students. Other areas not mentioned have been covered in parallel by both students.

1.6 Limitations

The concept of using GPUs in surveillance cameras is new. Thus, there is currently no hardware available for testing, not even on a prototype level. Performance measurements done on a different configuration than the targeted and this might be inaccurate and misleading. However, since the specifications of the target platform is well known, it is possible to choose a platform for performance measurements accordingly to give a proper estimation of the performance on the final platform. The targeted platform is limited to OpenGL ES 2.0 and GLSL 1.0. Thus, the final implementations will have the same limitations.

Using more than two camera sensors would allow to improve the result by removing some ambiguous correspondences from the matching phase and making it possible to produce a disparity map with wider and more accurate depth range. Although, because of the added complexity of three or more sensors, this report will focus on stereo cameras only.

Stereo calibration of the camera sensors will be considered out of scope since it is done once during the camera installation and not related to real time GPU computations.

Similarity, rectification of images, scaling and other operations suitable for the camera image pipeline will not either be covered by this thesis.

1.7 Acknowledgments

Mentors were Michael Doggett from the Department of Computer Science at the technical faculty of Lund University and Song Yuan at Axis Communications AB.

Chapter 2
Technology Background

2.1 History

Although the insight of how humans are able to receive two different images with the eyes is coming from the ancient Greeks, Sir Charles Wheatstone, in the 19th century, was the first to investigate the human stereo vision system in a systematic way. His invention, the stereoscope, is able to manipulate two pictures independently and present them to each eye simultaneously. A schematic of his invention can be seen in figure 2.1. In his first paper 1838 he was able to demonstrate that the disparities between two pictures in the stereoscope determined the relative depth in the picture, not the similarities in the pictures which was the angle of stereo vision experiments before Wheatstone [12]. With increasing disparity the viewer perceives objects in the pictures as closer. In the same paper, he also concluded that there were no eye movements involved when the depth was determined and that big differences in the two pictures resulted in conflicts and diplopia (double vision). Another interesting experiment Wheatstone presented in his paper was to show pictures of an object with different scales in the left and right picture. The viewer in the stereoscope saw the object in a scale in between the two sizes from the original pictures in the stereoscope[12]. The brain somehow interpolates the lines and presents a resulting average in between the two pictures in the stereoscope.

Human depth perception could be used as an initial approach for stereo vision. The human vision system use several depth cues combined to perceive depth.

There are several cues for extraction of depth information in a scene from only one image. It could for example be the aerial perspective method, which use the scattering of the light in the atmosphere which makes the contrast in an image reduce with the distance from an object in the scene. To be able to make this method useful, information about the atmosphere and the reflectance property of the object of interest must be known. This makes it a very uncertain and complicated method, thus not very reasonable to investigate further.

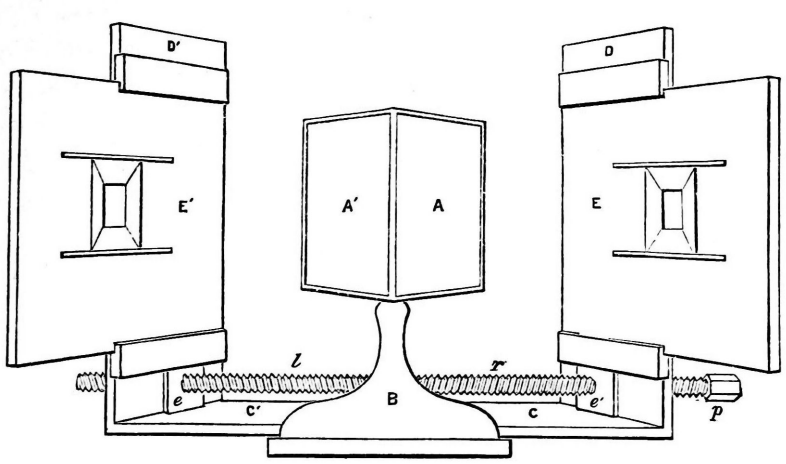


Figure 2.1: Sir Wheatstone's stereoscope (1838) [9]

An object that covers another object in the scene is perceived to be closer to the viewer than the occluded object. To be able to extract this information it has to be decided which surface that belongs to which object. The occlusion method can be a good way to find depth relative objects in the scene but the decision about what surface that belongs to what object is a difficult task. Moreover, it does not provide any information of the actual depth of an object, only the relative depth between objects.

The convergence cue involves using the relative orientation of the eyes when focusing on an object at a certain distance. If the viewing angle is large, i.e. the focus point of the eyes is close, an object is perceived as close. For a human, this would only work when the object is closer than approximately 6 meters to the eyes, objects at a longer distance makes the viewing angle too small to distinguish any difference in orientation[15]. Also, a camera setup using this cue as a base would require specific hardware which makes this cue impractical as a depth estimator.

Binocular disparity is the cue that has the largest contribution to human depth perception in comparison to other vision cues [40]. Depth in a scene is perceived as the difference between the images from the eyes. More disparity, i.e. more movement of an object between the images, indicates that an object is closer. The simplicity in positioning two camera sensors in parallel to mimic human binocular stereo vision has made it the most common solution for artificial depth perception.

2.2 Related Work

Using programmable shaders to implement a stereo algorithm for producing dense disparity maps is the focus of Karl Jonsson's thesis [35]. The objectives of the project is to study the advantages of GPU accelerating stereo correspondence with shaders in comparsion to a CPU implementation and to be able to use the result in real-time applications.

The study [41] presents a real-time stereo matching algorithm which is implemented in CUDA, a GPGPU library available for Nvidia GPUs. The goal of the study is to use depth information of a scene to place virtual objects. This means that the result from the stereo algorithm needs to be a dense, high quality disparity map. The algorithm, to be

able to accurately augment the scene, especially has to focus on the areas of the disparity map involving occlusion boundaries.

[38] is another previous study at Axis Communications AB. It investigates the possibilities of applying AR in surveillance camera systems. The study presents ideas for applications including for example "Creating a 3D world from real world". The idea involves identifying objects in the real world and replace them with artificial 3D-objects. The study also investigates properties in a single image that can be used to extract relative depth in a scene.

2.3 GPU

Computer vision and computer graphics are the opposites of each other in many ways. In computer graphics a three dimensional, numerical representation of a scene is projected onto a two dimensional display while in computer vision the input consists of two dimensional images which are used to find three dimensional, numerical models, as seen in figure 2.2.

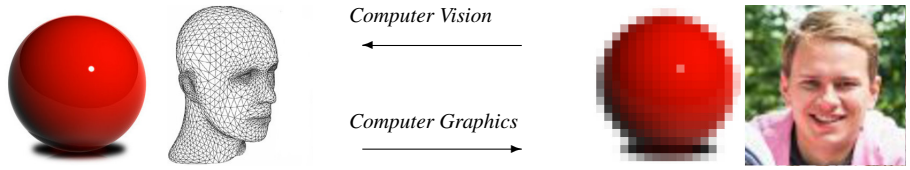


Figure 2.2: Computer Vision vs. Computer Graphics [2][6]

Although computer vision can be seen as the "inverse" of computer graphics, it is possible to take advantage of the highly data parallel arithmetic structure of the GPU originally used for computer graphics when implementing computer vision algorithms.

Most computer vision operations could be described as applying filter operations in sequences on images where the output from one filter is the input in the next. One single filter on an image could be seen as a SIMD operation which is very suitable for GPUs [39].

GPUs has been evolving constantly since the first simulations and games was developed in the mid 1970's. One of the first known GPU was used in an airplane simulator called "Whirlwind" for the US navy[45]. The GPU was at first designed for a small step in the graphics pipeline, transforming 3D coordinates to 2D screen coordinates. The basic idea was to distribute workload to several cores in parallel, making the computations run faster. When the development moved forward, more logic was placed in the GPU which made it possible to free even more CPU cycles. The first GPU that managed to implement the whole graphics pipeline was released in 1999 [36]. The card could handle various steps in the pipeline involving transformations, lighting, setup and clipping of triangles and rendering to the screen. Today, the graphics pipeline consists of four major steps [14].

1. Vertex Processing (Vertex Shader)

In the vertex shader, each incoming vertex is handled independently. The vertex

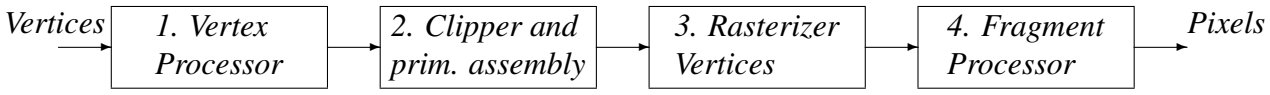


Figure 2.3: Graphics Pipeline

and its attributes are coming from the vertex buffer. After possible transformations the location of each vertex in 3D-space is passed further to next step in the pipeline.

2. Clipping and Primitive Assembly
This part of the pipeline computes which objects and what parts of the objects in the scene that are visible on the screen. To be able to make this decision, primitives need to be created based on the incoming vertices from the vertex shader. The output is the primitives in the scene whose projections are visible.

3. Rasterization
From the primitives in the output from the previous step, decisions about what pixels are inside a given polygon are done in this step. The output from the rasterization step is then a set of fragments for each primitive in the scene. The rasterization step also interpolates values from the vertex shader, such as colors, over the triangles.

4. Fragment Processing (Fragment Shader)
The fragment processing step runs independently for each pixel and outputs color values. The colors for each pixel are then drawn onto the frame buffer. The fragment processing is the most detailed and parallelizable part in the graphics pipeline.

The programmable pipeline has replaced the fixed function pipeline that was commonly used before. It uses programmable shaders e.g. vertex- and fragment shaders instead of having functions directly mapping to logic in the hardware which could only used for one purpose. Programmable shaders makes the GPU much more flexible and allows for more control over lighting- and material effects and vertex transformations.

As mentioned, GPUs consist of a big set of parallel and programmable cores. The evolution of GPUs the past decades has made it possible to use the GPU for general purpose computations instead of steps in the traditional graphics pipeline. The GPUs is now used in several areas outside primary graphics applications e.g. physics simulations, encryption and decryption, object detection, medical image reconstruction with FFT etc. [4]. General Purpose GPU (GPGPU) frameworks are written for this purpose. New type of shaders makes it possible to do computations on a GPU without being restricted to fixed inputs and outputs. In this way, the computations become even faster and more flexible.

OpenGL compute shaders are designed for exactly this purpose. There is no predefined inputs or outputs to the shader and it is not a part of the normal graphics pipeline. Instead, the compute shader could be viewed as a single-state pipeline. In the traditional pipeline, the fragment shader is executed separately for each fragment and the vertex shader executed for each vertex. To be able to parallelize the compute shader in a similar fashion, the computations is split into different parallelizable parts. These parts, called work items, is first grouped into local neighborhoods called local work groups. The local work groups are then grouped into a global work groups [18]. The global work groups can be seen as an entire frame in the traditional graphics pipeline and the local work groups are different groups of pixels in a frame connected to the same function. The input and output data for the compute shader is handled by the "Shader Storage Buffer Object". It is possible to read and write to the buffer freely within one shader, which makes it very flexible and suitable for other applications than graphics [11].

Chapter 3
Stereo Correspondence

The fast evolution of the GPU has made it possible to use it for efficiently speeding up complex and computationally heavy tasks within image processing and computer vision. One of these tasks is the stereo correspondence problem. Both image processing and computer vision utilize scene depth information for various applications. Furthermore, this is one of the fundamental parts of AR since it is necessary to correctly occlude and place objects in a scene that are going to interact with the real world. The interaction makes it important, not only to place the object at the correct depth, but also to receive information about the depth of the real objects in the scene. This application puts high demands on both precision and speed since it has to look realistic and be presented in real-time.

3.1 General Stereo problem

The general idea of stereo correspondence matching is to use two dimensional information from two or more images along with the position and calibration from the cameras to calculate three dimensional information of the scene [42]. Initially, this usually results in a disparity map, which essentially contains the distances between the pixels in the stereo images and not the actual scene depth.

3.1.1 Epipolar geometry

Two cameras with camera centers o_1 and o_2, where $o_1 \neq o_2$, aimed at the same scene form a relationship which can be exploited to estimate depth. The relationship is described by the fundamental matrix F, or the essential matrix E if the inner camera parameters are known, which maps a point p_1 in one image to a point p_2 in the other image and forms the epipolar constraint, shown in equation 3.1.

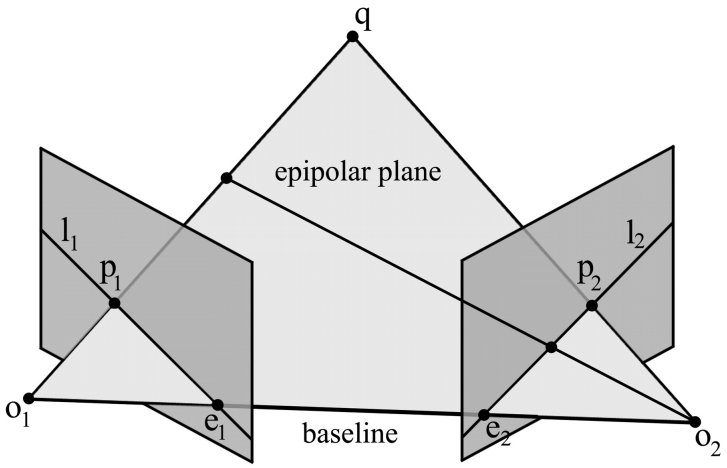


Figure 3.1: Epipolar geometry [1]

$$p_1^T F p_2 = 0 \tag{3.1}$$

The camera centers o_0 and o_1 together with a point q in the scene form the epipolar plane which intersects the image planes in the epipolar lines. As can be seen in figure 3.1 this restricts a point in one image to project onto a point along the epipolar line of the other image. Since the camera centers are known it is possible to use this relationship to triangulate the position, and consequently the depth, of q.

While it is possible to traverse the epipolar lines directly, even if they are diagonal, it is a somewhat impractical task. To make the task less complex it is possible to transform the images through a process called rectification to ensure that the epipolar lines are parallel and along the horizontal lines in the image, also known as scanlines. Traversing these scanlines instead of diagonal lines lowers the complexity and provides a more natural way of traversing the images as compared following a diagonal line. The relationship between disparities produced from a standard rectified geometry setup and the actual scene depth is extracted from formula 3.2, where d is the disparity, f is the focal length of the camera optics, B is the baseline, i.e. the distance between the camera sensors, Z is the actual scene depth and p is the pixel size.

$$d = f\frac{B}{Z * p} \tag{3.2}$$

3.1.2 Error analysis

Objective quality measurements of disparity maps is done with RMSE and BMP. The reason to use two different measurements is because the RMSE is a well known method to measure average error in the entire image while BMP measures how many pixels that are incorrectly matched. RMSE is defined in formula 3.3, where n is the number of disparities, d_f is the estimated disparity and $\hat{d}_f$ is the ground truth disparity.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(d_f - \hat{d}_f)^2} \tag{3.3}$$

BMP is defined in formula 3.4 where b is the decision function using the threshold t for determining if a disparity d is badly matched as compared to ground truth $\hat{d}$ and n is the number of disparities.

$$b(d, \hat{d}) = \begin{cases} 1, & \text{if } |d - \hat{d}| < t \\ 0, & \text{otherwise} \end{cases}$$
$$BMP = \frac{1}{n}\sum_{j=1}^{n} b(d_j, \hat{d}_j) \tag{3.4}$$

3.2 Global methods

Global methods focus on optimization of an energy function, a formulation of such can be seen in function 3.5, where C is a matching cost function, D is the disparity map and p is the current pixel. The second term in the energy function calculates a penalty value for all pixels with neighbors with a differing disparity value. All of the disparities in the stereo pair are computed simultaneously by using the optimization technique. Optimization on a global level is a NP-hard problem which makes it very difficult to use it in a real-time application, although it could result in higher quality disparity maps than local methods [30].

A wide range of approaches to the global optimization problem has previously been researched. Some of the more commonly used are graph cuts, belief propagation and dynamic programming such as [21], [17] and [20] respectively. Some advancements towards real-time global methods has been done, most of them are however barely real time and are accelerated with powerful desktop computer GPUs [19] [13] [26].

Apart from high computational costs, global methods usually has a very high memory usage due to their need to track global disparity information, which makes them even less suitable to use in a limited embedded system.

$$E(D) = \sum_{p}(C(p, D_p) + \sum_{q \in N_p} PT[|D_p - D_q| \geq 1]) \tag{3.5}$$

3.3 Semi-Global methods

Semi-global methods use ideas from both local and global matching to compute disparity with relatively good performance and low errors. The idea of SGM is to minimize the energy function 3.6.

$$E(D) = \sum_{p}(C(p, D_p) + \sum_{q \in N_p} PT[|D_p - D_q| = 1] + \sum_{q \in N_p} PT[|D_p - D_q| > 1]) \quad (3.6)$$

where the variable C, D and p is the same variables as in the global energy function 3.5. The penalties for small differences between the disparities is however treated differently than in the global methods. The function is minimized by approaching each pixel from several directions symmetrically and accumulating the costs for each disparity. The disparity with the lowest cost is then chosen as the winner. The method presented by [33] uses MI as the matching cost function, but Census has also been suggested as viable cost function [32]. SGM has similar drawbacks in memory consumption to global methods as they also keep track of global information. This is an issue in Hirchmüller's original presentation of SGM but solutions aiming to lower memory bandwidth and consumption has since been presented, one of them is eSGM [31].

3.4 Local Methods

Local methods try to find correspondences in a stereo pair by matching support windows. The support windows cover a squared neighborhood around the current pixel in each of stereo image and a matching cost is calculated and either minimized or maximized over the disparity range, depending the cost function. Since local methods work on isolated pixels without any global computations, calculations can be run in parallel which makes them very suitable to implement on a GPU.

input : A rectified stereo pair of the same size
output: A disparity map
foreach (x, y) *in left image* **do**
    `// Take the window around the pixel (x,y) in the left image.`
    $leftwin \leftarrow Left(x, y)$;
    `// For every disparity.`
    foreach d *in disparities* **do**
        `// Take the window around the pixel (x-d,y) in the right image.`
        $rightwin \leftarrow Right(x + d, y)$;
        $cost \leftarrow Cost(leftwin, rightwin)$;
        `// Greater than if we try to maximize.`
        if $cost < bestcost$ **then**
            $bestcost \leftarrow cost$;
            $bestmatch \leftarrow d$;
        end
    end
    $D(x, y) \leftarrow bestmatch$;
end

Algorithm 1: Pseudocode of the block matching algorithm.

The general local block matching algorithm is described in algorithm 1. To use a stereo pair for block matching the relative position of the two cameras has to be known

$$|z(t)| \leq 1 \text{ for all t}$$
$$|z(t)| = 1 \iff v(\tau) = \alpha * u(\tau - t0) \text{ where } \alpha \text{ is a constant}$$

Figure 3.2: Properties of NCC (Normalized Cross Correlation)

to determine in which direction the disparities should be traversed. The pseudocode steps to the left in the right image and stands on the same pixel in the left image but it would be equally correct to step to the right in the left image and stand on the same pixel in the right image. However, the resulting disparity map might differ depending on in which direction it is calculated, this is further covered in section 4.5.3.

There is several different local cost functions that can be used. The function chosen depends on a combination of performance, application and the quality expected from the disparity map. Table 3.1 contains the functions for all the cost functions described below where I_1 and I_2 are the input stereo images, (x, y) is the current position in the images, W is the support window, (i, j) is the current position in the support window and $\bar{I}_1$ and $\bar{I}_2$ are the mean values within the support window.

3.4.1 SAD

Sum of absolute differences is a very simple matching cost function but a fairly robust method since outliers will have a small influence on the cost. However, problems could occur when the method is used in real world applications since it does not compensate for differences in gain and exposure between the camera sensors, which is a very common issue.

3.4.2 SSD

local method Sum of squared differences has a higher computational complexity than SAD and is more sensitive to outliers which comes from the cubic term in the function. This is not a desirable property when the stereo algorithm is used on image from different kind of environments and with different kind of cameras and implies that the method is worse than SSD in images with noise and therefore less suitable to use in practice. Additionally, like SAD, SSD will have the same issues with differences in gain and exposure.

3.4.3 NCC

The standard Cross Correlation method is usually used to estimate the degree to which two series are correlated, but the 2D-version of the method is also able to identify patterns within images. It is therefore very suitable cost function for the local block matching method. More similarities between two areas result in a higher correlation value. The properties of Normalized Cross Correlation is presented in figure 3.2.

The Normalized Cross Correlation method is more computationally expensive than both SAD and SSD. As seen in table 3.1. The purpose of the method is to minimize the

SAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - I_2(x+i,y+j)\rvert$
SSD	$\sum_{(i,j)\in W} (I_1(i,j) - I_2(x+i,y+j))^2$
NCC	$\frac{\sum_{(i,j)\in W} I_1(i,j)\cdot I_2(x+i,y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1(i,j)^2 \cdot \sum_{(i,j)\in W} I_2(x+i,y+j)^2}}$
ZSAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i,y+j) + \bar{I}_2(x+i,y+j)\rvert$
ZSSD	$\sum_{(i,j)\in W} (I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i,y+j) + \bar{I}_2(x+i,y+j))^2$
ZNCC	$\frac{\sum_{(i,j)\in W}(I_1(i,j) - \bar{I}_1(i,j))\cdot(I_2(x+i,y+j) - \bar{I}_2(x+i,y+j))}{\sqrt[2]{\sum_{(i,j)\in W}(I_1(i,j) - \bar{I}_1(i,j))^2 \cdot \sum_{(i,j)\in W}(I_2(x+i,y+j) - \bar{I}_2(x+i,y+j))^2}}$
LSAD	$\sum_{(i,j)\in W} \lvert I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i,y+j)}\cdot I_2(x+i,y+j)\rvert$
LSSD	$\sum_{(i,j)\in W} (I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i,y+j)}\cdot I_2(x+i,y+j))^2$

Table 3.1: Matching cost functions [3]

influence of differences in exposure and lightning conditions in the two images.

3.4.4 Zero-Mean Methods

The zero-mean versions of the local cost functions compensate for a constant offset in gain by using the mean result from both the left and the right image. Differences in the two images is a case where SAD and SSD typically will not be able to give a satisfying result, the addition of zero-mean will reduce this issue. In zero-mean NCC the combination still have the purpose of decreasing the influence of differences in gain between the two images but are using more information and therefore could be even more stable against this kind of problems. Zero-mean methods work well when the variance of the pixel values is small, which leads to small differences between pixel values and the corresponding mean values in the images.

3.4.5 Locally-Scaled Methods

The locally scaled methods use a relative term to scale the value in the second image. The term is the ratio of the mean values computed from the two images. This will decrease errors based on constant differences between the images. This works better than the zero-mean methods when the pixel values have more variance, which leads to bigger difference between the pixel values and the corresponding mean values. Because of the design of the NCC cost function, it is not possible to scale only one image with a term.

$$\begin{bmatrix} 41 & 154 & 111 \\ 201 & (100) & 56 \\ 12 & 231 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 1 \\ 1 & X & 0 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow 01110010$$

Figure 3.3: Census transform of a 3x3 block

$$\begin{bmatrix} & 154 & \\ 201 & (100) & 56 \\ & 231 & \end{bmatrix} \rightarrow \begin{bmatrix} & 1 & \\ 1 & X & 0 \\ & 1 & \end{bmatrix} \rightarrow 1101$$

Figure 3.4: Mini- Census transform of a 3x3 block

3.4.6 Census Transform (SHD)

Census is a cost function just as the functions described above, but differs completely in the way the cost is calculated. Instead of using the original values, all pixels in the support window are encoded into a binary bit value representing the relationship between the current pixel and the pixel in the center of the support window. The bit is set to one if its value is higher than the center pixel, otherwise the value is set to zero as shown in figure 3.3. After encoding, the Hamming distance between the two bit vectors from the encoded support windows in the left and right image is calculated and used to compare the similarity between the images. The Hamming distance between two vectors with equal length is, by definition, the number of coefficients in which they differ, i.e. the number of bits that has to be changed in one vector to make the vectors equal [5]. By using the Census method, it is possible to get a disparity map between two pictures from cameras with completely different internal properties. Problems occurring with other cost functions due to imbalanced brightness will be eliminated. Thus, the Census algorithm is considered to be the most robust cost function, even though the method is sensitive to high frequency noise [44].

It is possible to save computational power by reducing the length of the bit vector used for calculating the Hamming distance in the Census Transform method. This can be done by only using a part of the block window for the computations, as shown in figure 3.4. This method is called Mini-Census.

3.4.7 Hierarchical Block Matching

In [43] Tanimoto and Pavlidis suggests a hierarchical method to increase search space in image analysis by scaling down the image first, start on a low resolution and work towards the original image resolution in a so called image pyramid. In this way a large part of the image can be covered in the high levels of the pyramid and then the lower levels helps to refine the initial result from the higher levels with a possibly lower computational cost than if the same area of the image would be searched in full scale.

Local block matching methods, such as the methods described above, cover a limited part of the image which depends on disparity range and support window size. Increasing the disparity range and window size to cover larger areas of the image is computationally expensive and grow even larger with increased image resolution since larger windows and disparity ranges has to be used to cover the same area of the image. Using a hierarchical approach, such as the method described by Koschan et al. [16], makes it possible to cover larger areas of the images with small support windows on high levels of the pyramid which could help in difficult areas where the support window does not reach enough details on full scale to find a unique match. As proposed by Jonsson [35] it is possible to reach a lower complexity with hierarchical block matching than standard block matching if the disparity ranges on low pyramid levels use a fixed range rather than an exponentially growing range as originally proposed by Koschan et al. [16].

The first step of hierarchical block matching is to create an image pyramid by scaling down the original images until the desired number of pyramid levels is reached. Scaling can be performed with a Gaussian kernel such as function 3.7 where x and y is the relative position from the (0,0) centered pixel and σ is the variance.

$$G(x, y) = \frac{1}{2\pi\sigma} e^{\frac{-(x^2+y^2)}{2\sigma}} \tag{3.7}$$

Hierarchical block matching starts at the highest level of the pyramid, i.e. the lowest resolution, for which the initial disparities is computed. This step has no input since there is no higher level which makes this step unique. It works like standard block matching and traverses disparities only in one direction. The initial disparities could then scaled up by interpolation [35] and used as input to the next level in the pyramid which searches disparities in the immediate neighborhood, in both directions of the initial disparity. The second step is then repeated for every pyramid level until all levels are traversed and the final refined disparity map is the output of the lowest level.

3.4.8 Evaluation of Cost Functions

As this study targets real world scenes the chosen method has to be able to handle situations that might not be covered with tests made on standard stereo pairs. These images are often created under perfect conditions with ideal camera settings. However, a real world scenario might introduce difficulties which some cost functions does not handle well. It is reasonable to test the cost functions for such situations. The test consists of two different parts where the right image of the stereo image pair remains the same but where the left image is distorted in different ways. The purpose of the test is to investigate how stable and well suited the different cost functions are when there is unexpected differences between stereo images. In a real situation, these differences might be due to specularities in the scene or different gain in camera sensors or other effects that is very likely to occur in a real world set up.

The evaluation of cost functions is done by calculating disparity maps based on two different image pairs for all local cost functions except Census Transform since GLSL 1.0 shaders does not support bit operations.

In the first part of the test the left image is distorted by multiplying every pixel with a constant value. The value 1.9 was chosen arbitrarily to represent a plausible gain and

luminosity difference that could occur due to the difficulty of configuring two camera sensors equally. In the second part the left image is distorted with a uniformly random pixel-wise offset in brightness. A value is chosen in the range between 1.7 and 2.1. It is then, similar to the first part, multiplied with the value of its corresponding pixel. It simulates a different kind of distortion in the camera but also involves differences in brightness between the two images. This part of the test is designed mainly to find differences between the zero-mean methods and the locally scaled methods which both compensates for an offset in brightness but in different ways. Both tests are made on distorted versions of the Tsukuba stereo pair.

3.4.9 Result and Analysis

Table 3.2 and 3.3 show that SAD and SSD gave significantly better results when the zero-mean and the locally scaled methods were applied. This suggests that it might be suitable to use these methods with the addition to either the zero-mean or the local scaling method.

When the zero-mean and locally scaled methods were compared with SAD and SSD, the zero-mean methods gave the best result in both parts of the tests. The reason for this result in the first part is straightforward since the zero-mean method is exclusively designed to compensate for a constant offset in brightness between the two images. That was exactly how the first part of the test was designed, creating a constant offset by multiplying every pixel with a constant. The second test differs from the first because of a pixel wise variance in the offset in the second image. The reason to apply this kind of noise was because of the difference between the scaled and zero-mean approaches described in section 3.4.4 and 3.4.5, however, the zero-mean approach still gave a better result than locally scaled. This could be because of the variance on the pixel noise in the second image was not big enough to make the locally scaled approach more preferable than zero-mean. It is possible that a higher variance in the pixels would result in the locally scaled approach giving a better result than the zero-mean.

The standard NCC method worked better without adding zero-mean. The standard NCC method also gave a better overall result than any other cost function on both parts of the test. Therefore this is the cost function that will be used in further investigations in the study.

	RMSE	BMP
SAD	5.6198	0.64992
SSD	5.5403	0.62866
NCC	3.3702	0.3055
ZSAD	3.7732	0.35803
ZSSD	3.7358	0.34828
ZNCC	4.0609	0.32242
LSAD	3.8386	0.36034
LSSD	3.9367	0.37482

Table 3.2: Constant offset in brightness (1.9) in the left Tsukubua image. 5x5 support window size with 0-16 disparity range.

	RMSE	BMP
SAD	5.6193	0.6489
SSD	5.5402	0.62887
NCC	3.6636	0.35485
ZSAD	4.1285	0.41088
ZSSD	4.0024	0.38978
ZNCC	4.3986	0.38017
LSAD	4.5234	0.45546
LSSD	4.4601	0.44521

Table 3.3: Random pixel wise offset in brightness (1.7-2.1)in the left Tsukubua image. 5x5 support window size with 0-16 disparity range.

Chapter 4
Refinements

4.1 Support window size

The size of the support window is very critical [34]. In order for the algorithm to be able to match the pixels in a correct way, the window has to be big enough to cover enough intensity, or color variations. On the other hand, a too large window leads to undesirable smoothing effects and a disproportionately large foreground in the disparity map. Smaller support windows lead to a faster computation of the disparity map, but makes it more difficult to find unique matches which leads to noisy output.

4.2 Confidence Metrics

All the local cost functions described above are strong in detailed areas and weak in areas with repetitive patterns or with only one single color. In such weak areas several disparities would produce scores that come close, or are equal, to the best score. Thus, it could be very helpful to mark these difficult areas and use the information about which disparities that are of low confidence for post processing or for applications that use the disparity map.

4.2.1 Uniqueness of minimum/maximum

There might be pixels in the stereo images for which the local methods are not able to find a unique disparity match. The main reason for this is because of similarities between matching windows at different disparities, as described above. One method to decide if the resulting disparity value is of high or low confidence is to keep track the three best matching scores [37]. By investigating the difference between the best and the worst of these three scores it is possible to decide if a match is of high or low confidence. The

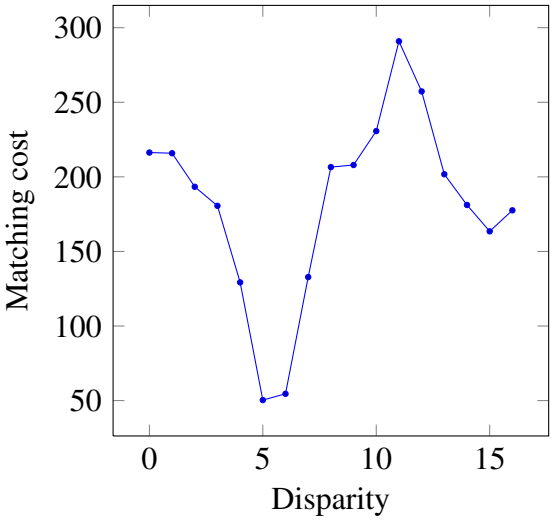


Figure 4.1: Typical matching score distribution of a good match.

rationale behind tracking three values is that it is possible to have a double minimum, but not a triple minimum or more. Since disparity matching initially results in discrete disparities while the actual depth is continuous the real disparity is always in between two disparities, which makes every match a double minimum, as seen in figure 4.1. Thus, the confidence of the match with respect to uniqueness is not related to the second best disparity score [37].

4.2.2 LR-RL consistency check

Occlusions occur when one of the camera sensors captures a surface which is occluded to the other camera sensor. Since such pixels only exist in one of the images the matching step will fail to find the correct pixel and produce erroneous disparity values.

Ideally, calculating the disparities from left to right should be equal to calculating them from right to left. However, due to occlusions and other difficulties, this is seldom true. By calculating two disparity maps, one for each direction, and comparing their disparity values it is possible to find which resulting disparities that are likely to be of low confidence due to occlusions.

4.3 Filters

Local stereo correspondence usually produce a combination of salt-and-pepper noise over the whole disparity map and very noisy and erroneous disparities in areas where matching is difficult, such as in surfaces without texture. A way to minimize these side effects is to apply filters to the disparity map.

4.3.1 Median filtering

An approach to remove salt-and-pepper noise is to use a median filter with a small kernel, which is an effective way to remove single pixel or very small areas of noise but does not improve larger erroneous areas. Choosing a median filter to filter disparity maps can be motivated by its edge preserving abilities and that it outperform Gaussian smoothing on the kind of noise usually introduced by local methods [10]. Another important property of a median filter is that it does not introduce new values as a mean filter could do. Furthermore it is possible to use weighted median filters to improve the filtering as in [24] and [29].

4.3.2 Bilateral filtering

Bilateral filters has many of the desired properties suitable for filtering disparity maps such as that it is edge preserving, noise reducing and has smoothing effects [28]. The idea of bilateral filtering is to weight pixels in the neighborhood not only by Euclidean distance but also by radiometric differences such as color distance or intensity difference. Bilateral filtering can be defined as in formula 4.1 where I_{out} is the filtered result, I is the image to be filtered, x is the location of the currently pixel, Ω is the window centered in x, f_r is the range kernel and g_r is the spatial kernel.

$$I_{out} = \frac{\sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(x_i - x)}{\sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(x_i - x)} \tag{4.1}$$

The range kernel produces weights that depend on radiometric differences and the spatial kernel depends on the Euclidean distance. The kernels are smoothing functions where usually Gaussian functions are used, presented in 3.7. A depth discontinuity in a scene is often a discontinuity in luminosity and/or color in the scene, which is best described by the original stereo images rather than in the disparity map. This property can be exploited by using one of the original stereo images as input to the range kernel instead of the disparity map, this is called a cross-bilateral filter. One issue with a bilateral filter with Gaussian kernels is that it produces smoothing in a similar way that a mean filter would do, hence introducing new disparities. Instead, it is possible to use a bilateral filter as a weighted median filter by simply taking the median of the weights. This may be efficiently implemented with a histogram approach and used together with the confidence metrics, as the ones introduced in sections 4.2.1 and 4.2.2, to create an adaptive cross-bilateral median filter, such as the one proposed by Müller et al. [23].

4.3.3 Mathematical morphology

Mathematical morphology is a technique commonly used to extract features and refine digital images. It works with a set of operators which can be combined to alter the image in different ways depending on what the goal is. The basic set of operators is erosion, dilation, opening and closing. Each operator use a structuring element which is a binary image which will match similar shapes in the image. Thus, the structuring element is commonly chosen to match common shapes in the image. Erosion, formula 4.2, is the

process of going through all the pixels in the original image X and applying the structural element S on the neighborhood of each pixel. From the values covered by the structuring element the minimum is chosen to replace the current pixel. Dilation, formula 4.3, is the opposite process and takes the maximum within the structuring element. Morphological opening, equation 4.4, of an image is done by first eroding the image and then dilating the result of the erosion. Closing, equation 4.5, is done by first dilating and then eroding [25].

$$X \ominus S = \bigcap_{s \in S} X_{-s} = \{x \in \mathbb{Z}^2 | S_x \subseteq X\} \tag{4.2}$$

$$X \oplus S = \bigcup_{s \in S} X_s = \bigcup_{x \in X} S_x = \{x + s \in \mathbb{Z}^2 | x \in X, s \in S\} \tag{4.3}$$

$$X \circ S = (X \ominus S) \oplus S \tag{4.4}$$

$$X \bullet S = (X \oplus S) \ominus S \tag{4.5}$$

While median and bilateral filters could be used to remove noise and smooth out poorly matched areas they will not fill more than very small gaps. Schmidt et al. [22] proposes the use of a morphological closing operator to fill larger gaps. However, the possible size of the gaps filled is limited to the size of the structuring element. While larger structuring elements fills larger gaps they also remove more detail, so this trade off has be taken into consideration while selecting a suitable structuring element. Because of the computational complexity, the techniques presented in this section will not be investigated further.

4.4 Gap filling

It is very hard to create reliable, completely dense disparity maps solely through local block matching methods. To be able to create disparity maps with as high density as possible, a way to handle low confidence pixels in an efficient way is essential. While filters such as the median filters described in section fill out small gaps and smooth out bad areas, they produce very limited or no results at all when gaps grow larger than the filtering kernel. Furthermore, while increasing the kernel size would let the filter fill larger gaps, detailed areas will be unnecessarily smoothed. Instead of filtering the disparity map further, it is possible to attempt to fill out the gaps [27] [22]. Occlusions found with a LR-RL consistency check may be filled in with the lowest disparity on either side of the gap, since an occlusion is often caused by a foreground object occluding a background object [27].

However, this assumption alone is not sufficient to remove all gaps. Another type of gap occurs when matching is difficult due to very small score/cost variations between disparities. This may be due to radiometric issues such as over or under exposure, single color surfaces or repetitive patterns. Unfortunately, to fill those gaps with correct disparity

is a much more complex task than filling occlusions.

```
input  : A rectified stereo pair of the same size and a disparity map
output: A gap filled disparity map
foreach s ← scanline in disparity map do
    gaps ← FindGaps(s);
    // For every disparity.
    foreach g ← gap in gaps do
        startEdge ← FindEdge(left, g.start);
        endEdge ← FindEdge(left, g.end);
        // If there is an edge on both sides, we have no
           best guess and leave the disparities untouched.
        if not startEdge and not endEdge then
            Interpolate(D, g);
        end
        else if startEdge and not endEdge then
            Fill(D, g, D(g.end + 1));
        end
        else if endEdge then
            Fill(D, g, D(g.start − 1));
        end
    end
end
```

Algorithm 2: Pseudocode of the gap filling algorithm.

By extracting features from the original stereo pair and use them as hints it could be possible to fill out some of the undefined parts of a disparity map. Performing edge detection on each side of a gap could make it possible to make assumptions about what the surroundings of the gap look like. Algorithm 2 use the idea that gaps could be filled by looking at the surrounding edges in the original images. The idea is as follows: if there are edges on both side of the gap it is likely that the surface is either a foreground surface surrounded by two background surfaces or a background surface surrounded by two foreground surfaces, since this case has no best guess the disparities should be left untouched. No edges on either side indicate that the gap is in the middle of an evenly textured/colored surface, thus it has to be interpolated between the disparities on both sides of the gap to account for surfaces which are not parallel to the camera direction. A single edge on one side of a gap indicates a discontinuity in depth at that side which makes it more likely for the disparity on the opposite side to be correct, much like in the case of an occlusion.

4.5 Results and Analysis

4.5.1 Support window size

In table 4.1 the errors are presented from disparity map calculations with different support window sizes. For the tests the SAD method was applied on the Tsukuba stereo pair. The choice of support window sizes tested was made depending on guidelines from other

	RMSE	BMP	FPS
3x3	6.1715	0.44092	410
5x5	4.6461	0.29291	276
7x7	3.8512	0.23608	164
9x9	3.4126	0.2087	4

Table 4.1: SAD method used on the Tsukuba stereo pair with different support window sizes.

researches and on the performance of the previous window size during the test. When the 9x9 support window size test reported 4 FPS on the Tegra board, which would result in speeds way below real-time an embedded GPU, no further testing was needed. As presumed the performance decreased with larger support window sizes. However, the drastic performance penalty going from 7x7 windows to 9x9 implies something that had not been a problem with smaller windows suddenly became a bottleneck, a possible reason could be lack of cache hits in the texture cache of the GPU. The optimal support window size depends both on what application the disparity map will be used for and on what resolution the source images have. A high resolution needs a larger support window to get an equivalent result to a smaller support window on a lower resolution. Based on the performance results, the conclusion was that it seemed reasonable to use support windows sizes of 5x5 or 7x7.

4.5.2 Uniqueness of minimum/maximum

Figure 4.2 shows the result of the uniqueness of maximum method when NCC with 5x5 support window size was used. The red areas in the figure show that some of the larger, lesser detailed areas in the image were found. The results confirms the idea that detailed non-repetitively textured areas of the images are uniquely matched and very few pixels in these areas are marked. Since both large and small areas is marked with this method it could be reasonable to use it both for weighting filters and to aid filling large gaps. Moreover, the simplicity of this method combined with its effectiveness and accuracy makes it very suitable for further use.

4.5.3 LR-RL consistency check

The result from applying the LR-RL consistency check on the Tsukuba stereo pair is presented in figure 4.3. While the main purpose of the LR-RL consistency check is to find occlusions it can be seen in the figure that it also marks disparities which clearly are not in occluded areas. This is most likely due to the unpredictable behavior of local stereo correlation in areas without textures, which result in high frequency errors. Thus, these areas will unlikely be matched equally in both directions, which explains why they are marked as inconsistent. Checking for consistency is a very robust way of finding occlusions and other bad matches since it does not rely on thresholds or attributes of the scene. However, it is rather expensive since it more than doubles execution time.

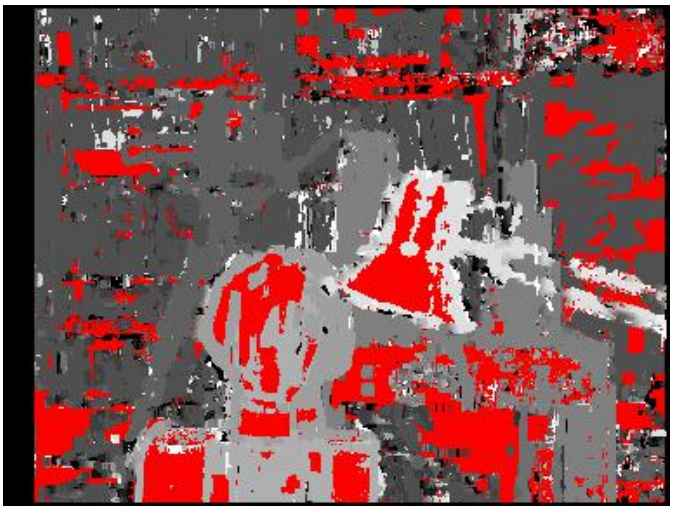


Figure 4.2: Red areas show low confidence pixels found with the uniqueness of minimum/maximum method.

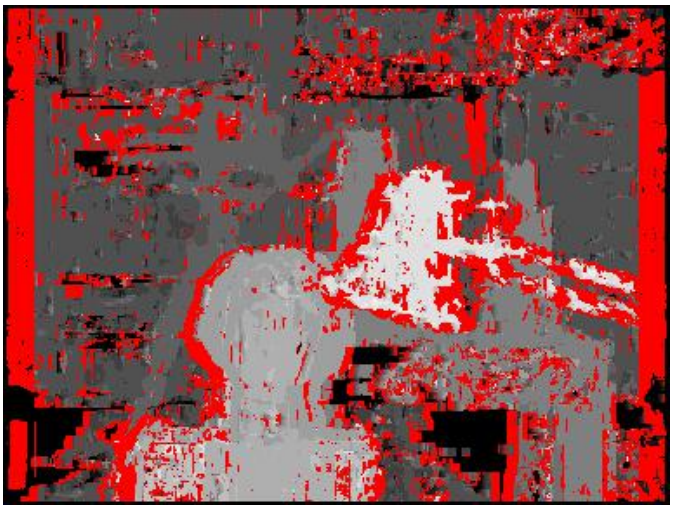


Figure 4.3: Red areas show low confidence pixels found with the LR- RL consistency check.

4.5.4 Filters

Tables 8.4, 4.4 and 4.5 show the results from filtering disparity maps computed with NCC with a window of 3x3 and 5x5 respectively, which each produce a decent amount of noise. The test were also rerun including gap filling. The error improvements from small kernel median filters compares very well to the improvements from the more advanced cross-bilateral median filter, even after several iterations the difference to the median filter is very small.

4.5.5 Gap filling

Results from gap filling is combined with the results from filtering in tables 8.4, 4.4 and 4.5. In the Tsukuba stereo image pair gap filling works very well and lowers the BMP error by approximately 42% and 35% for the 3x3 and 5x5 original disparity maps respectively. While this might be seen as very good results it should be noted that Tsukuba have very favorable circumstances, e.g. a complete lack of gaps with edges on both sides, which made the filling work very well.

	RMSE	**BMP**
Unfiltered	2.6952	0.23258
Median 3x3	2.4171	0.19973
Median 5x5	2.2076	0.17332
WCBiMF 5x5 (1)	2.2071	0.16924
WCBiMF 5x5 (2)	2.1909	0.16317
WCBiMF 5x5 (3)	2.1884	0.16103
WCBiMF 7x7 (1)	2.1281	0.15804
WCBiMF 7x7 (2)	2.12	0.15359
WCBiMF 7x7 (3)	2.1258	0.15305

Figure 4.4: Results from filtering tests.

	RMSE	**BMP**
Gap Filled, Unfiltered	2.1636	0.15251
Median (3x3)	1.8833	0.12737
Median (5x5)	1.7004	0.10938
WCBiMF 5x5 (1)	1.7174	0.10983
WCBiMF 5x5 (2)	1.6891	0.10573
WCBiMF 5x5 (3)	1.6882	0.10474
WCBiMF 7x7 (1)	1.65	0.10272
WCBiMF 7x7 (2)	1.639	0.09995
WCBiMF 7x7 (3)	1.6442	0.09935

Figure 4.5: Results from filtering test where the gaps of the disparity map was first filled.

Chapter 5
Investigations

5.1 Disparity Distribution

A confidence metric that emerges from the uniqueness of minimum/maximum idea presented in section 4.2.1 is to examine how the minimum/maximum values from the two best matching scores are distributed. Instead of only observing the third best score this metric use the insight of how the matching scores possibly are spread out over the disparities in a bad pixel and the assumption that more distribution in disparities between the two best matching score values would indicate that the match is ambiguous and therefore the confidence of the match is lower. Two almost equal matching scores could be a result from a repetitive pattern in the scene, something that would not be found with the standard uniqueness metric. Figure 4.1 shows the matching scores of a pixel that are in a detailed area, thus easily matched, in the stereo images. In this graph the two smallest matching scores are next to each other, a possible double minimum, but if the two best scores would more spread out over the disparity range the match would not contain a double minimum.

The metric is then used with a threshold to determine the confidence of a match. The most intuitive value of the threshold would be one (1) since the two best scores should be next to each other. In other words a pixel match has low confidence if the second best matching score's disparity is not next to the best disparity.

5.2 Number of Active Pixels in Support Window

Block matching, as described in section 3.4, use square windows where all pixels within the support window are included in the calculation. The exception is Mini Census which uses a cross pattern instead of the full square. The idea of this section is to study how the

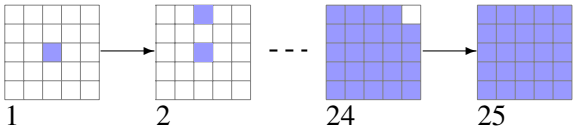


Figure 5.1: An example of the number of active pixels in support window test. First, second, 24th and last iteration.

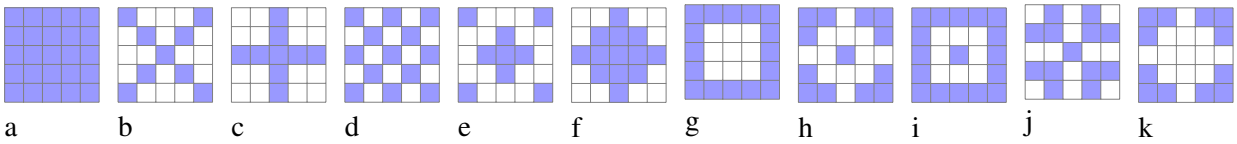


Figure 5.2: Patterns in support window

number of pixels used within the support window affects the results. Intuitively the errors should decline relative to the number of pixels in the window.

5.3 Patterns

The results from the "Number of active pixels in support window" investigation 8.4 suggests that there is a correlation between the number of active pixels in the support window and the error in the resulting disparity map. However, since error improvements declined very rapidly it could be merit to not use the full window to save calculations. To investigate further how the design of the support windows affect the result, different patterns are tested which should give an idea of how the distribution of active pixels affects the results.

In section 5.2 a 5x5 support window space base is used when filling in with active pixels. The same size is used for picking the patterns shown in figure 5.2. The particular, the layout of the patterns tries to cover different parts of a 5x5 window with different pixel densities. All patterns in the investigation have between 9 and 17 active pixels, which is motivated by the results from table 8.4. The gains up until around 9 pixels is considered significant enough that is no reason to try patterns with less than 9 pixels, likewise the quality gains above 17 pixels is considered too small, as shown in figure 5.6.

To minimize the influence of the scene properties, the test is done on 3 difference stereo pairs. The known Tsukuba stereo pair, the Moebius stereo pair shown in figure 8.5 and the Dolls stereo pair shown in figure 8.6. The reason to make this investigation is not only to find out if this kind of modification results in a disparity map with higher quality but also to try to increase the performance of the algorithm. Fewer active pixels in the support window size lead to fewer calculations which should lead to better performance. The aim is to find a pattern that both minimize the number of active pixels and the errors in the resulting disparity map.

5.4 Every Other Disparity Match with Neighbor Control

A way to improve the performance of block matching could be to take two disparity steps instead of one when going through and calculating the matching scores for a pixel, then calculate the values of the neighboring disparities of the best match from the first step to find the final disparity. The idea is based on the understanding of what a typical graph of matching scores look like. Figure 4.1 is an example, where it is clear that this idea is something that could be considered when trying to increase performance. The best match for the pixel represented in figure 4.1 has neighbors with almost as good score as the best match. In this case the minimum value is guaranteed to be found since either the best match will be found directly, or one of its neighbors will be found and then the best match will be found when controlling the score of the neighbors.

5.5 Results and Analysis

5.5.1 Disparity Distribution

A first test was tailored to distinguish the differences between this method and the uniqueness metric. Two images was created artificially and used as stereo pairs. Both images are based on the same striped pattern with 5 pixels wide black and white stripes. An artificial stereo pair showing a striped surface, i.e. a repetitive pattern, is created by moving the pattern in the right image one pixel to the left. By finding low confidence matches in this scene with both the uniqueness and disparity distribution metric, the differences between the methods could be measured. The result of the method is shown in figure 5.3. A disparity difference of more than one in a pixel was considered bad and is marked in red. Compared to the uniqueness metric, also shown in figure 5.3, it is clear that the disparity distribution metric marks more pixels as bad. The uniqueness metric finds 10% of the pixels while disparity distribution metric finds all of the low confidence matches in the image.

Figure 5.5 shows the result of setting the threshold value to 1 and 6 with the disparity distribution metric on the Tsukuba stereo image pair. A smaller threshold results in a significant part of the image being marked. Even though the natural approach was to set the threshold to one, it could be reasonable to evaluate if a bigger value is more suitable.

The conclusion is that disparity distribution metric is able to find areas in the image where patterns become a problem for the disparity matching and where the uniqueness metric does not help since it is mainly extracted from the second best match. The metric should not be considered a separate confidence metric but rather an improvement of the uniqueness metric since it misses some of the critical low confidence areas which are found with uniqueness.

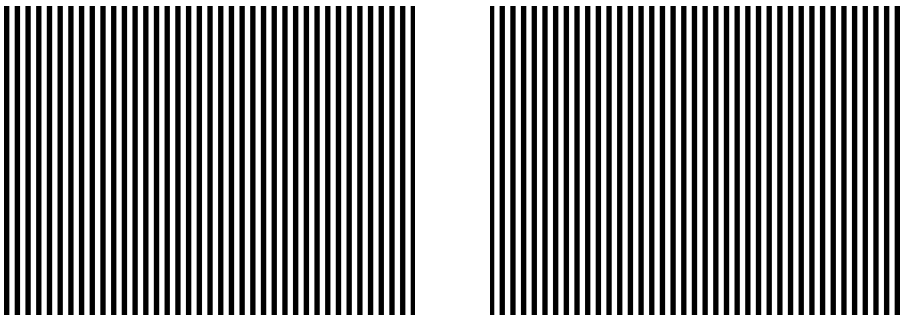


Figure 5.3: Striped pattern test image for comparison of between the uniqueness and disparity distance metrics

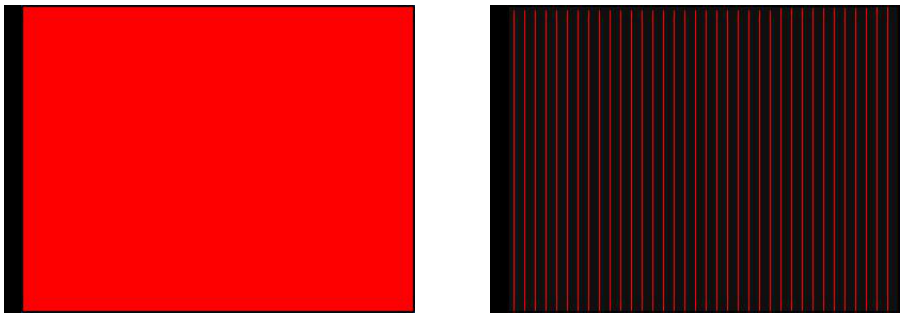


Figure 5.4: Left: Result "Disparity Distribution" Right: Result "Uniqueness of minimum/maximum", low confidence matches are marked with red.

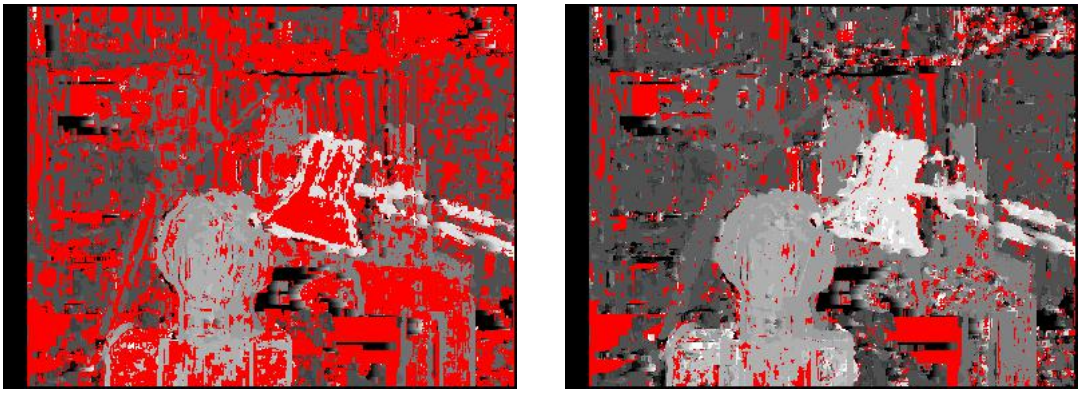


Figure 5.5: NCC method with 5x5 support windows used. Red areas show low confidence pixels found with the disparity distribution method with threshold value 1 (left) and 6 (right).

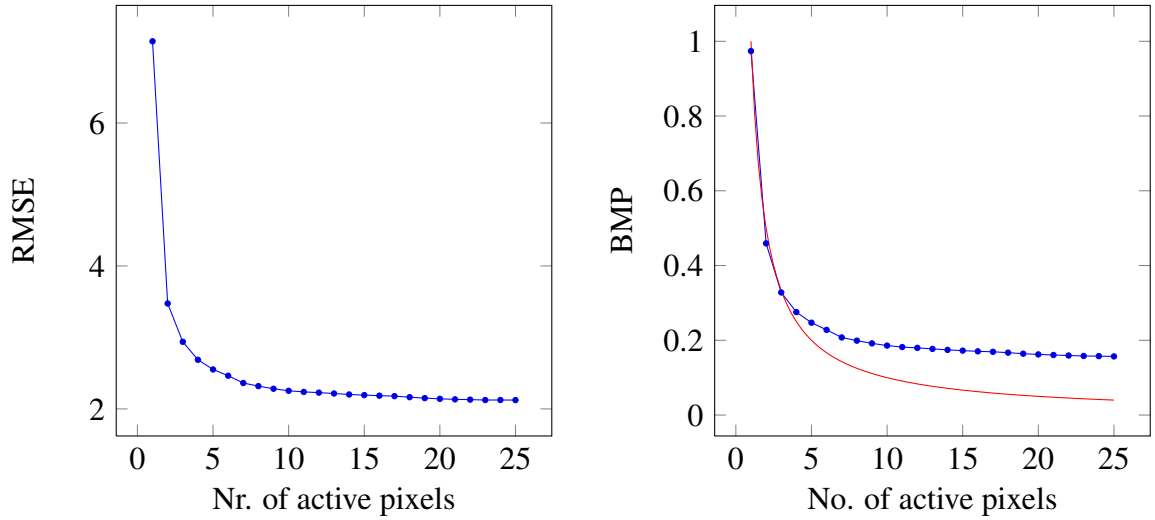


Figure 5.6: Plots showing how the RMSE and BMP errors behave with a different amount of active pixels in the support window, the BMP error is compared to the function f(x) = 1/x (red).

5.5.2 Number of Active Pixels in Support Window

Starting with the center pixel of a 5x5 sized support window, additional pixels was added with a uniform distribution inside the window until all pixels was filled in. After a new active pixel had been set, a block match was performed on the Tsukuba stereo pair and errors was calculated. This was repeated until the whole window was filled in. Figure 5.1 show how the pixels was added in the first and last iterations of the test. The test was repeated five times and the result was averaged to minimize the influence of particular distributions.

As expected, having just a few active pixels in the support window results in large errors and more active pixels gives a smaller error. However, the error improvement for each added pixel declines very fast, faster than the relative number of pixels, which can be seen in figure 5.6. The error plots errors reveals that the error improves greatly up until around 10 active pixels, then adding more pixels results only in marginal improvements. The main goal of this investigation was to find if the error decline was relative to the number of pixels within the window, the results suggest that this is not the case since the error decline in all cases is less than or equal to what the relative contribution of an added pixel should be.

5.5.3 Patterns

The cross pattern performed worse on average than the other patterns that was using nine active pixels, which shows that not only the number of pixels but also the location of the active pixels affects the quality of the resulting disparity map.

To strengthen this statement a comparison was made between the edges pattern errors 8.1 and the errors in a disparity map, where a random pattern with the same amount

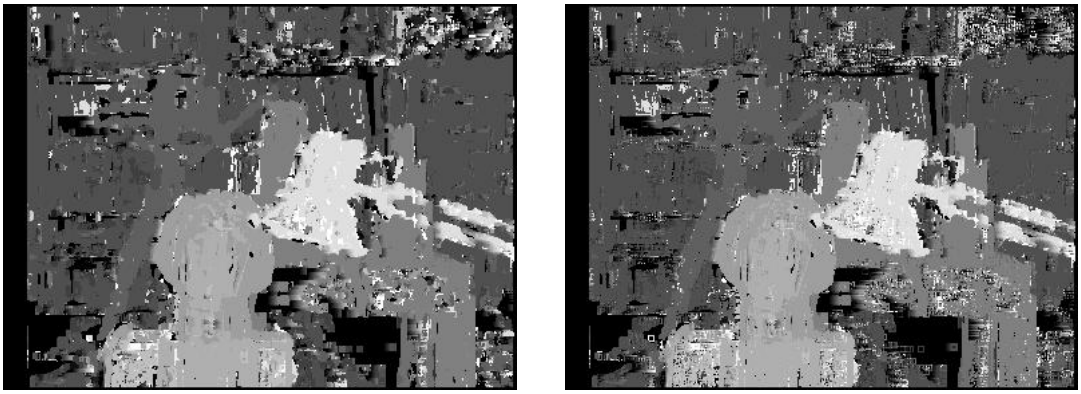


Figure 5.7: NCC method with 5x5 support window size with no pattern (left) and with the edges pattern (right)

	RMSE	BMP
b. Cross (9)	1.0616	1.7628
c. X (9)	1.0383	1.6823
d. Uniform (13)	1.0369	1.6857
e. Corner (9)	1.035	1.6728
f. Diamond (13)	1.033	1.6656
g. Edges (16)	1.0167	1.6204
h. 3 Corner (13)	1.0193	1.6299
i. Edges + c (17)	1.0168	1.6187
j. Turned corners (13)	1.0335	1.6701
k. Inv Diamond (12)	1.0201	1.6298

Table 5.1: Mean difference in RMSE and BMP from the original no-pattern disparity map in percent from the three images. The three best scores are marked.

of active pixels as the edges pattern were used, presented in table 8.4. Both tests were done on the Tsukuba stereo pairs. By comparing the results, which are very different, a conclusion was drawn that the resulting disparity map actually depends highly on where the active pixels within the support window are placed.

The resulting disparity maps in figure 5.7 show the results both when edges pattern and full pattern are used. A conclusion was drawn that most of the differences between the two images are present in poorly textured areas and single color areas. This is a desirable result since these areas will be marked as bad through the uniqueness metric presented in section 4.2.1 and could be handled in a different way than the high confidence pixels, for example sorted out or filtered harder.

A conclusion from the result in table 5.1 was that it is very suitable to use the g, i or k pattern from figure 5.2, instead of full support windows. They will not result in better quality than a full support window, but could be considered good trade-offs for performance over quality.

	RMSE (5x5)	BMP (5x5)	RMSE (9x9)	BMP (9x9)
Two Step	3.757	0.36926	3.0604	0.24389
Two Step + Neighbor Check	3.7126	0.3598	2.9924	0.23318
Original	2.8261	0.23576	2.9322	0.19721

Table 5.2: Results from checking every other disparity with 5x5 and 9x9 support windows.

5.5.4 Every Other Disparity Match with Neighbor Control

The results in table 5.2 suggest that there is merit to only checking every other disparity. If there was no correlation between the score of the neighbor to the best match the errors would have been much larger since half of the matches would have been arbitrary. Still, it is also apparent that it will decrease the quality of the resulting disparity map, i.e. it is still strictly better to traverse all disparities. The importance of controlling the neighbors of the disparity with the highest score is however not obvious from the table, but becomes clear by observing the resulting disparities in figure 5.8 where large areas will be one disparity from the correct disparity without the neighbor control. The quality penalty with 5x5 support window is almost 53% in BMP but using 9x9 window reduces the relative penalty to 18% which suggests that this method works better with increasing window size.

The resulting images for the 9x9 support window are shown in figure 5.8 together with ground truth [8] used for error calculations.

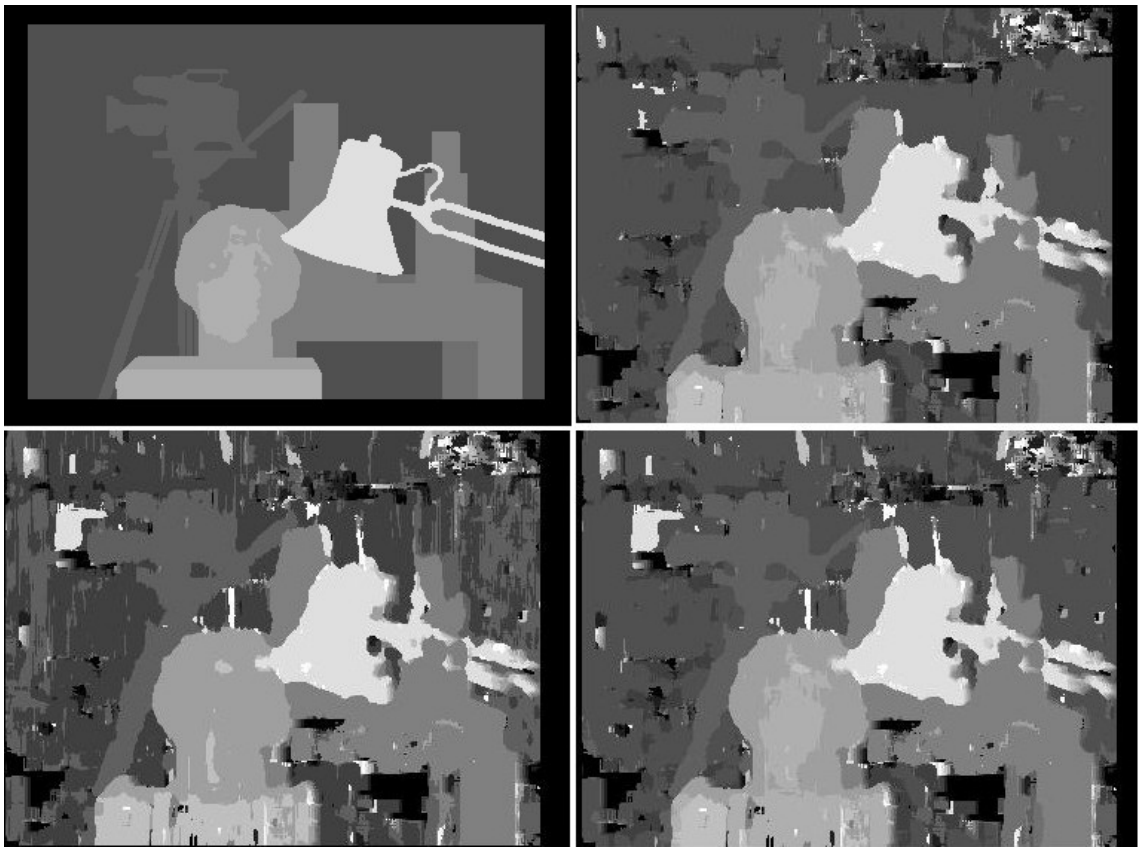


Figure 5.8: 9x9 support window two step disparity check result. Top left: Ground truth; Top right: Without two step; Bottom left: Two step; Bottom right: Two step + Neighbor check

Chapter 6

Implementation

6.1 Platform

6.1.1 OpenCV

With no actual stereo camera hardware available some of the image pipeline parts had to be simulated. This was done by using OpenCV, which is a software library for computer vision and machine learning. More specifically the parts needed for the simulation were: camera calibration, stereo calibration, stereo rectification, image operations/transformations and background subtraction. OpenCV includes all this functionality with a similar work flow as Matlab or Octave.

To produce a good stereo calibration it could be favorable to calibrate each sensor separately first. By sampling images from each sensor with a visible calibration pattern, in this study a 9x6 chessboard was used, OpenCV computes the intrinsic parameters of each sensor. These parameters only depend on the inner configuration of the cameras, such as sensor and optics. Stereo calibration is then conducted in a similar fashion with the same calibration pattern. In this step the pattern has to be visible in both sensors simultaneously. Taking the intrinsic parameters from the individual calibration as an initial guess, the stereo calibration produces a new set of intrinsic parameters together with a set of extrinsic parameters. The extrinsic parameters describe the sensors relative configuration. The procedure was repeated until a satisfactory result was achieved, an RMSE and re-projection error below 1.0.

The resulting intrinsic and extrinsic parameters were later used as input to the stereo rectification, also in OpenCV, which transform and warp the images so that the epipolar lines are parallel, a necessary step for the stereo block matching algorithm to work properly. OpenCV was also used to capture images from the camera sensors, scale them to the correct size and transform them into grayscale.

NCC	5x5 (full)	5x5 (edges)	5x5 (full) LR-RL	5x5 (full) gap filling
Unfiltered	230	250	121	48
Median (5x5)	215	226		
WBM	26			

Table 6.1: Performance for the Tsukuba stereo pair on the Tegra K1 board in frames per second.

NCC	5x5 (full)	5x5 (edges)
Median (5x5)	115	155

Table 6.2: Performance for the live implementation on the Tegra K1 board in frames per second.

6.1.2 OpenGL

OpenGL is a library for rendering 2D and 3D graphics in real time. It takes primitives as input and produces 2D raster images as output. In the past, the graphics pipeline was fixed, which meant that lighting calculations and similar operations was totally controlled by the pipeline, out of reach for the application programmer. However, newer versions of OpenGL uses a programmable pipeline, described in figure 2.3 where the programmer is allowed to write custom shaders for manipulating vertices and writing color data to fragments.

The graphics library used in the implementation is OpenGL ES 2.0 along with shading language GLSL 1.0. While this is not the latest version, it is the latest version that is available for the targeted GPU.

This study uses the programmable pipeline in a non-conventional way to exploit the computing power of the GPU, similar to how the GPGPUs are used. Instead of producing an image from a set of primitives, an image, more specifically a disparity map, is produced from two other images which are uploaded to the GPU as textures. The only primitive is a full screen quad which is used as a canvas for rendering the disparity map, as shown in figure 6.1.

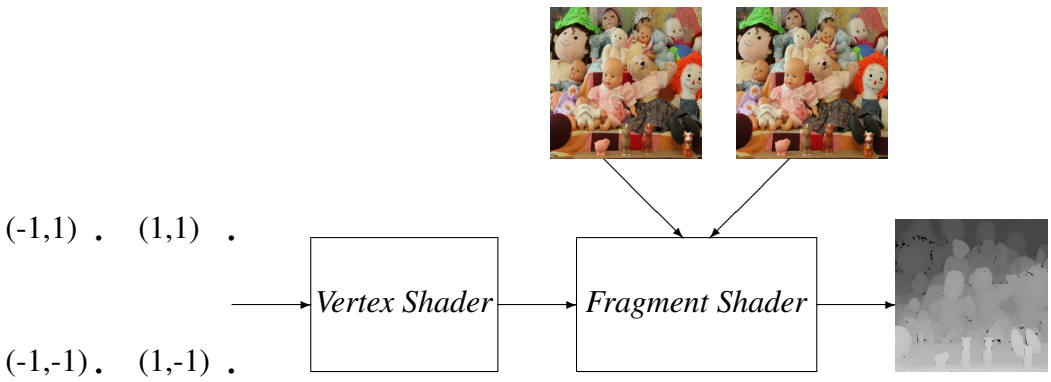


Figure 6.1: The data flow of the implementation.

6.2 Implementation of Stereo Matching

The final stereo correlation system is described in figure 6.2. Based on the results from section 3.4.9, NCC was chosen as the matching cost function for the local stereo correlation. It is possible to chose between using the sparse support window in NCC to get a better performance or to use the full support window. The input also includes disparity range. The output from the NCC correlation shader is a texture with four 8bit channels, in other words a regular RGBA texture. Disparity values are stored in the first channel, the red channel, and the following three are reserved for confidence metrics.

To measure the confidence of a match the final algorithm include two mandatory confidence metrics based on uniqueness of the best match and disparity distance between the best matches. Optionally a LR-RL consistency check can be included which is implemented in a separate shader.

The filtering part of the algorithm is proposed to use a median filter with a kernel size of 5x5. This is motivated by its relatively low complexity and that the results was basically on par with the much more advanced weighted cross-bilateral filter. An other alternative is to use the cross-bilateral filter, which is significantly slower than the median filter, but gives a better result in some cases. Additionally, if even more quality is wanted, it is possible to iterate the cross-bilateral filter more than one time. It is recommended to run the LR-RL check if the cross-bilateral filter is used since it uses that metric as a weight. Lastly, if speed is highly prioritized over quality or filtering is made in further processing of the disparity map, it is possible turn off filtering altogether by using a filter shader that leaves the disparity map unaffected.

Figure 6.3 shows an example input image and two resulting images from the final algorithm, one from sparse support windows and one when full support windows are used. The resulting disparity maps have utilized the uniqueness metric to find sort out low confidence matches. These pixels are black in the result.

In table 6.1 and 6.2 the performance of running the final algorithm on Nvidia Tegra K1 is presented. The tables show the performance of the algorithm using different filters and the sparse/full pattern. The performance tests were both done with the standard Tsukuba stereo pair as input and with a live stream.

6.3 Optimizations

6.3.1 RGBA used as YYYY

It is possible to reduce the number of texture fetches by packing luminance values into RGBA, four at a time, which in theory could reduce the number of texture fetches needed to a fourth compared to using a luminance texture. However, since the values now are packed into vectors a lot of index computations has to be done to fetch and use the correct values. The idea is not investigated further in this study.

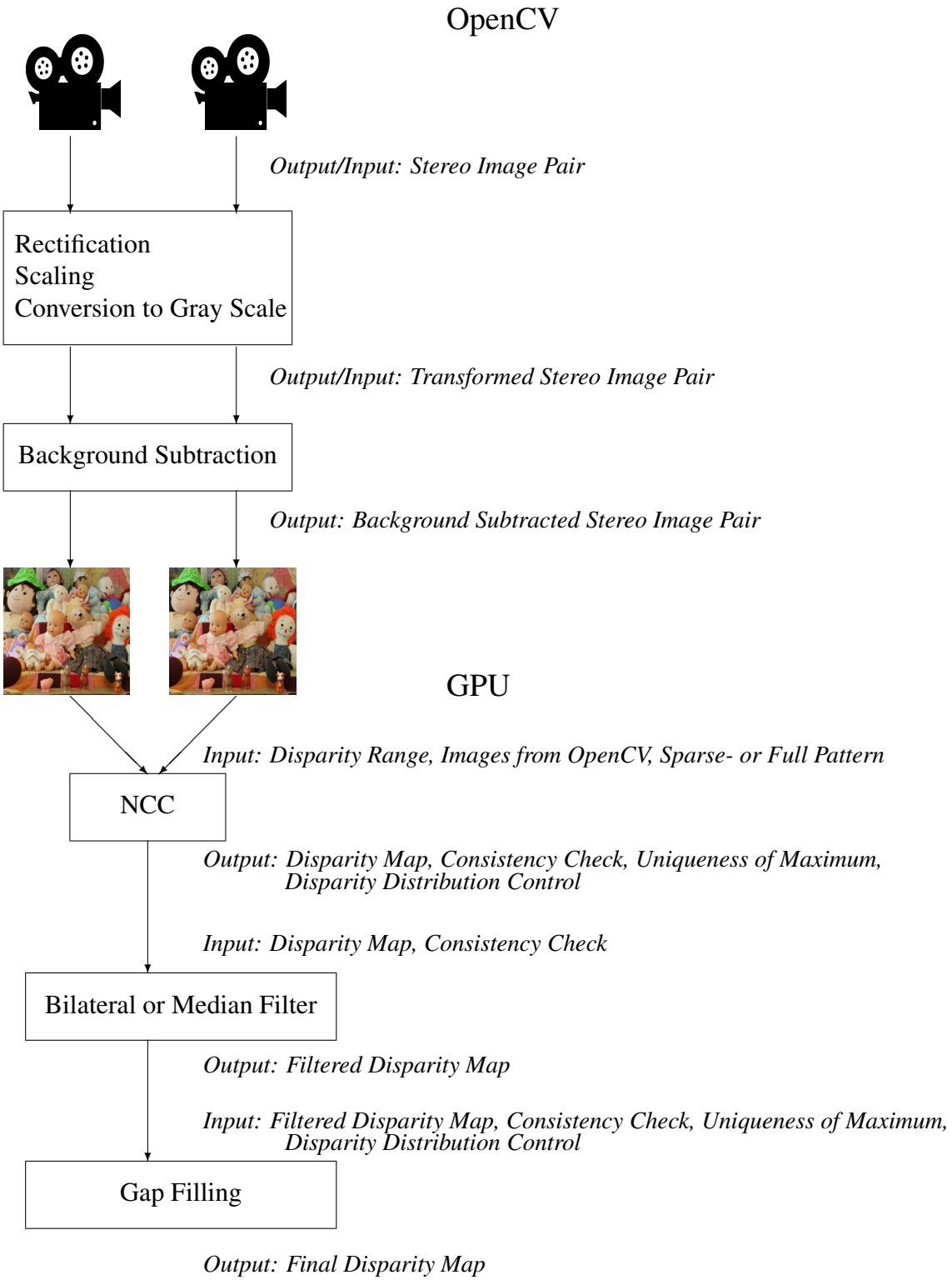


Figure 6.2: The final algorithm

a

b

c

Figure 6.3: Result when using NCC with 5x5 support window size, 0-16 disparity range and uniqueness of maximum control. a, original image b, full window c, sparse window

6.4 Video Surveillance Applications

Possible applications for disparity maps range from real time AR to video compression. As a demonstration, a couple of applications were described and implemented.

6.4.1 Flexible speed measurements

With a single camera it is possible to measure the speed of objects if the camera is mounted perpendicular to the direction of the objects, for example measuring car speed by having the camera mounted on the side of the road. The downside of this solution is that it restricts mount positions of the camera which might severely limit the surveillance coverage. With a stereo camera setup it would be possible to mount the camera at an angle that cover much more of the scene while still being able to estimate the speed of objects by adding depth to the calculations. The information about the speed of an object is not only useful in traffic applications but can also be good when identifying an object in a scene.

6.4.2 Depth mask

In some cases video surveillance has to be restricted to a certain area in the images. This might be due to limitations in the video surveillance permission or that some sensitive information might be captured. Current solutions use a manually defined area in the video that is masked out. While it blocks out the sensitive information in an effective way, it may also block out objects or persons, which might be of interest, in front of the sensitive area. If the depth of an object in a scene is known, it is possible to define which objects that are in front of or inside the restricted areas.

6.4.3 Object size estimation

A common feature in surveillance cameras is motion detection and ability to track detected objects. A detected object is often encapsulated inside a bounding box and/or polygon which could give an operator a rough idea of the object size. However, without any real perception of the depth in a scene, it is very hard to estimate the real size of a detected object. By using the area of the bounding box or, preferably, a polygon around an object together with the underlying disparity map it is possible to estimate the real size of the object. While the value has no real unit it give a representation of the object size and reasonable thresholds could be used to determine, for example if an object is a human or something smaller or larger. If the real object size is required it would be possible to convert the disparities into depth in meters with formula 3.2. One motivation for determining the actual object size could be to avoid false alarms where small animals such as cats or birds manage to come close to the camera and appear as large objects.

Chapter 7

Discussion and Conclusion

7.1 Discussion

While the field of research of disparity maps and similar topics is very active and has been for a while, a lot of the focus is towards quality rather than performance. This is apparent by taking a look at the top algorithms at the KITTI stereo benchmark site [7]. Some of the methods takes almost an hour to complete which makes them unsuitable for real time applications. This might be to a difference in focus between the industry and academic research, where academic research might tend to go for quality over performance. In some cases it is essential to produce a dense, high quality disparity map, but in other application an approximation of the disparity map would be good enough. One of these cases are when the disparity map is benefited for video compression. Different applications require different quality and performance. This is why one of the aims in the study was to make the algorithm as flexible as possible.

One of the bottlenecks of a shader solution lies in the massive amount of texture fetches needed for each window match. Furthermore there is no way in conventional OpenGL shaders to reuse the texture fetches between fragment shader runs which means that there will be a lot of overlap. In OpenGL compute shaders, which are available from OpenGL version 4.3, it is possible to write and read arbitrarily to texture memory and also share variables between shader instances. These features fits very well and may be exploited to reduce the overlap significantly and thus lower graphic memory requirements and reduce execution time. This is mainly why compute shaders, and general compute languages, is so powerful when doing stereo correspondence. Due to the limitations of the targeted hardware, compute shaders were not investigated in this thesis. This has been used in several other studies, such as [41].

7.1.1 Comparison to Related Work

Karl Jonsson [35] draws the conclusion that it is indeed more suitable to implement local stereo correlation on a GPU than on a CPU. Additionally the quality of the resulting disparity maps on even limited GPUs can be competitive with the disparity maps calculated on CPUs when taking performance into consideration. This study started where [35] ends. After an initial realization that GPUs are very suitable for stereo match algorithms, the study went deeper into different, completely new ideas of optimizing both performance and quality of the stereo algorithm in GPUs.

While [41] investigates the possibilities to use stereo matching algorithms to create real time AR applications on laptop- or desktop computers, this study presents methods for the same kind of applications but for very limited GPUs in embedded systems, for example surveillance cameras. This leads to a deeper investigation in GPU-specific performance optimization and a flexible algorithm that is adjustable depending on the design of the embedded system. The insight that the stereo matching is fundamental for almost every AR application is used in both studies to argue for the chosen subject; stereo matching algorithm for GPUs to produce disparity maps in real-time.

Although there is a lot of similarities between this study and [44], for example both studies investigate the possibilities to implement stereo algorithms on an embedded system, the choice of hardware differs and affects the areas investigated. Because of the choice of implementing the stereo algorithm on FPGA, a big part of [44] describes a hierarchical method to optimize performance of the stereo algorithm. The implementation of the same method in a GPU would not be as straight forward because of the inability to share memory, but the ideas investigated in this study can be applied to a hierarchical method easily and is something that is left for future work.

Jonas Nolte and Magnus Wetterberg [38] describe AR ideas for surveillance camera systems. Most of the applications are based on the assumption that it is possible to obtain information about the relative depth in the scene. This study's main focus is not designing AR applications for surveillance cameras systems. It concentrates instead on optimizing the fundamental stereo algorithm that makes the AR applications possible. The stereo camera is never mentioned in [38]. Alternately, the study examine the possibilities of perceiving depth from the scene by considering features in only one image from a single camera.

7.2 Conclusions

One of the goals for the study was to design an algorithm for computations of disparity maps that can be tweaked easily and modified to fit the needs of the application and limitation of computational complexity. The programmable pipeline of OpenGL ES 2.0 forces solutions to be modular and easy to modify through shaders. Thus, the final algorithm may be modified freely by tweaking constants inside the shaders or by switching shaders altogether.

It was considered probable that quality of the disparity maps would have to be compromised to be able to reach the goals of real time, or near real time, speed on the targeted hardware. The aim was to compromise as little quality as possible to gain as much perfor-

mance as possible. Primarily, two novel ways of compromising quality for performance was proposed: sparse window matching and every other disparity matching with neighbor control.

Sparse edge window matching increases the frame rate significantly with 35%, by reducing the number of operations in the matching stage and by exploiting the vector computation capabilities of the GPU, while most of the quality decrease is introduced in areas where matching is difficult even without the performance modification. Another approach to increase the performance was to traverse every other disparity and then check the neighbors of the best disparity match. In this way the amount of operations needed was was reduced further, however, depending on the scene this may result in a moderate to severe quality reductions which makes this method less potent than sparse edge window matching.

Since it was expected and well known that the resulting disparity maps will have flaws in certain areas, ways to find and mark those disparities was considered as important in the design of the algorithm as increasing performance. The uniqueness constraint in combination with the disparity distribution control is used to find low confidence matches to help post processing and applications. Together with a consistency check that computes the disparity map the opposite way most of the bad matches can be found and processed differently for example with gap filling.

7.3 Future Work

All of the implementations and performance tests were done on development boards, instead of the actual targeted hardware. Thus the first step after the thesis would be to implement and rerun tests on the forthcoming Artpec 6 chip when it is available.

While the focus of this study was to find ways to speed up local window correlation it is further possible to use these ideas along with methods that use local methods in a possibly more efficient way. Using sparse edge window matching along with a hierarchical method such as the one used in [35] or [44] could lead to performance gains and produce disparity maps with higher quality. If a wider disparity range or higher resolution is demanded, hierarchical solutions would probably be an interesting approach.

In a similar way to hierarchical methods, SGM could also take advantage of the sparse support windows modification to increase performance of matching cost calculations. This could as well be a natural step forward to extending this study.

An insight during the thesis was that it would be possible to achieve a wider depth range by using more than two horizontally parallel cameras. In many, for example outdoor applications, a normal stereo camera setup could be a limitation. In these situations a solution with more cameras is suitable. Optimizing the relative placement of the cameras and design further tests with this kind of setup would be an interesting and useful investigation.

It could be reasonable to extend the local method test in section 3.4.9 with more than two stereo pairs. The reason for the limitation of the test was because of the outstanding result from the NNC method. Although it could be interesting to investigate further in how the locally scaled and zero-mean methods differs if the variance in noise was higher.

An interesting application that was never implemented during this study is the flexible

speed measurement described in 6.4.1. Details involving how to actually calculate object speed information in a 3D scene are left for future work.

Chapter 8
Appendix

	RMSE	BMP
a. No pattern (25)	2.1235	0.15685
b. Cross (9)	2.3567	0.20269
c. X (9)	2.2281	0.18301
d. Uniform (13)	2.2261	0.17686
e. Corner (9)	2.2455	0.18429
f. Diamond (13)	2.2886	0.18930
g. Edges (16)	2.1214	0.16172
h. 3 Corner (13)	2.1392	0.16555
i. Edges + c (17)	2.1269	0.16046
j. Turned corners (13)	2.2113	0.17675
k. Inv. Diamond (12)	2.1397	0.16605

Table 8.1: 5x5 patterns applied on Tsukuba stereo pair.

	RMSE	BMP
a. No pattern (25)	1.2172	0.027369
b. Cross (9)	1.2321	0.041359
c. X (9)	1.2132	0.037444
d. Uniform (13)	1.2222	0.039671
e. Corner (9)	1.2001	0.036654
f. Diamond(13)	1.2446	0.042526
g. Edges (16)	1.2165	0.038845
h. 3 Corner (13)	1.2121	0.038019
i. Edges + c (17)	1.2133	0.03854
j. Turned corners (13)	1.2172	0.039473
k. Inv. Diamond (12)	1.2168	0.037929

Table 8.2: 5x5 patterns applied on Moebius stereo pair.

	RMSE	BMP
No pattern	1.3159	0.02877
Cross	1.3987	0.071494
X	1.4069	0.072266
Uniform	1.3927	0.07135
Corner	1.3971	0.072051
Diamond	1.3890	0.071422
Edges	1.3840	0.069357
3 Corner	1.388	0.070345
Edges + c	1.3845	0.06977
Turned corners	1.3939	0.070237
Inv. Diamond	1.3857	0.070345

Table 8.3: 5x5 patterns applied on Dolls stereo pair.

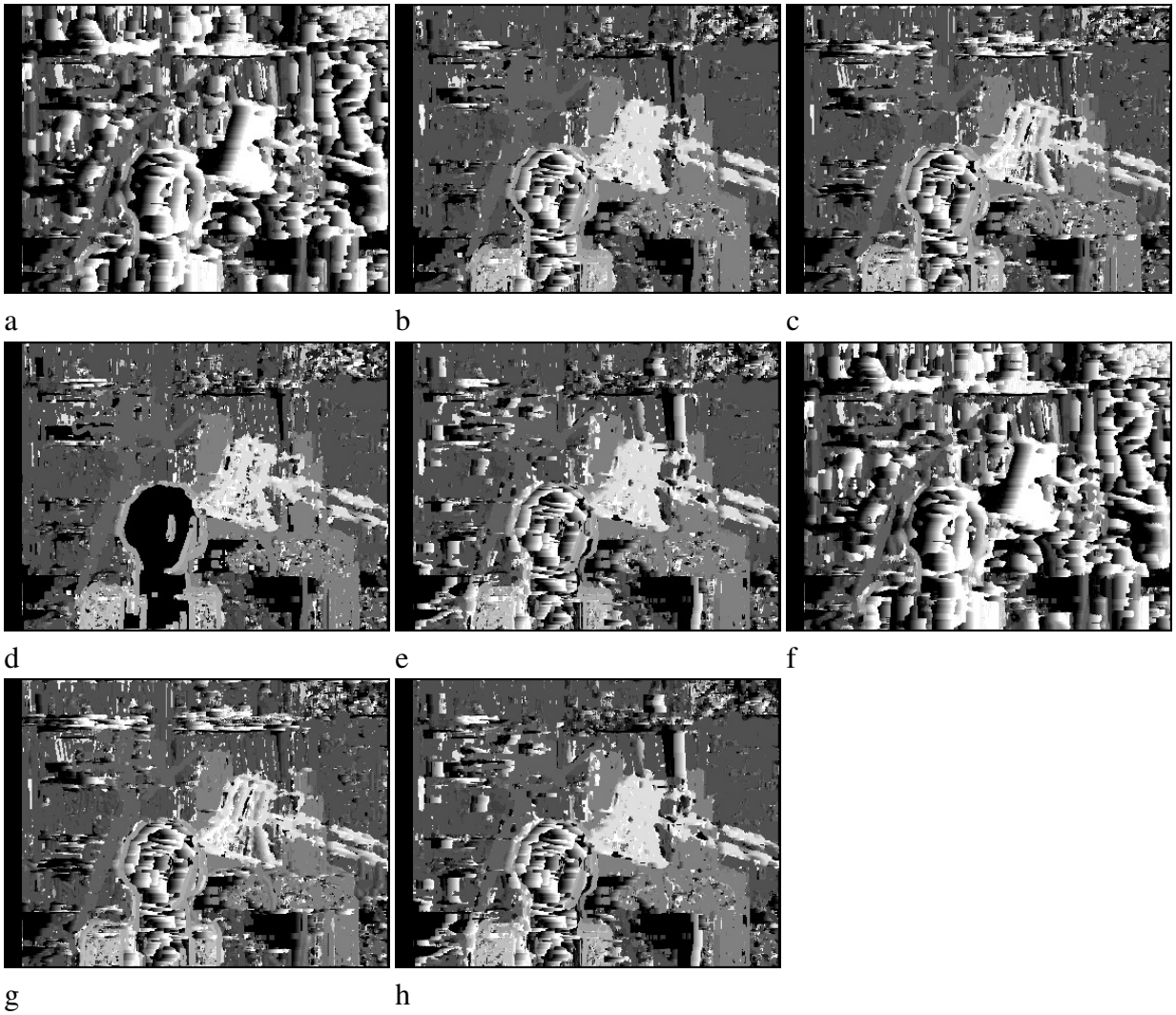


Figure 8.1: Result with 5x5 support window size, 0-16 disparity range and a constant offset in brightness (1.9) applied to the left Tsukuba image. a, SSD b,NCC c, ZSSD d, ZNCC e, LSSD f, SAD g, ZSAD h, LSAD

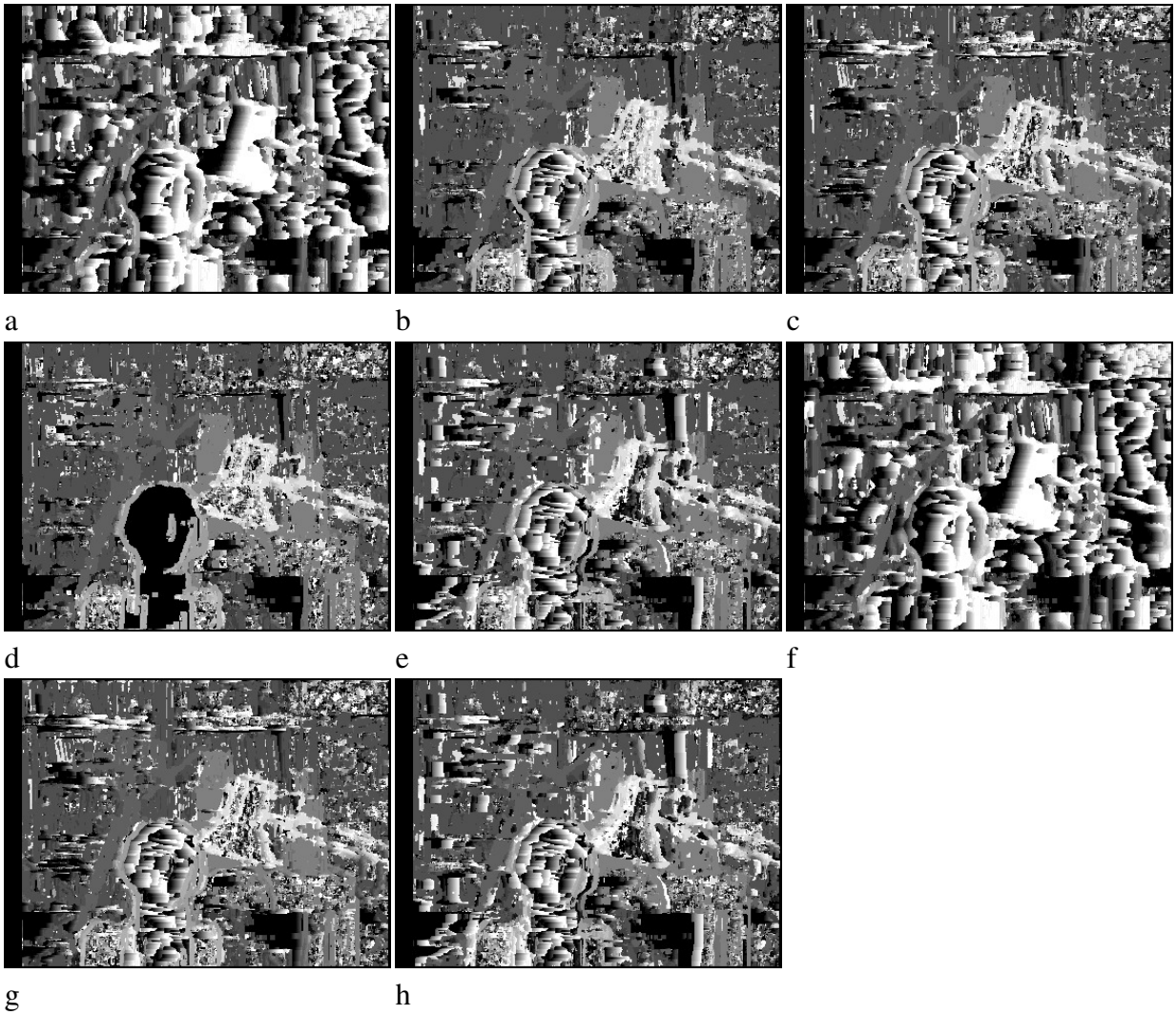


Figure 8.2: Result with 5x5 support window size, 0-16 disparity range and a random pixelwise offset in brightness (1.7-2.1) applied to the left Tsukuba image. a, SSD b,NCC c, ZSSD d, ZNCC e, LSSD f,SAD g, ZSAD h, LSAD

	RMSE	BMP
1	7.143	0.97385
2	3.475	0.45931
3	2.9391	0.32796
4	2.6876	0.27543
5	2.5522	0.24711
6	2.4645	0.22786
7	2.3627	0.20764
8	2.3189	0.19902
9	2.2821	0.19177
10	2.2538	0.18584
11	2.2381	0.18189
12	2.2279	0.17974
13	2.2159	0.17697
14	2.2019	0.17436
15	2.1934	0.17236
16	2.1854	0.17043
17	2.1787	0.16925
18	2.1641	0.1669
19	2.151	0.16413
20	2.1412	0.16219
21	2.1339	0.16062
22	2.1299	0.15926
23	2.1245	0.15807
24	2.1248	0.1576
25	2.1235	0.15685

Table 8.4: Average results from each step in the number of active pixels test.

	RMSE	BMP
Unfiltered	3.378	0.36667
Median 3x3	2.7235	0.28078
Median 5x5	2.4397	0.23672
WCBiMF 5x5 (1)	2.5479	0.23478
WCBiMF 5x5 (2)	2.5433	0.22784
WCBiMF 5x5 (3)	2.5503	0.22615
WCBiMF 7x7 (1)	2.4886	0.2243
WCBiMF 7x7 (2)	2.501	0.22122
WCBiMF 7x7 (3)	2.5155	0.22084

Figure 8.3: Result from median and WCBiMF filtering of a disparity map from NCC with 3x3 support window size.

	RMSE	BMP
Gap Filled, Unfiltered	2.4807	0.21478
Median 3x3	1.9709	0.15894
Median 5x5	1.7216	0.13467
WCBiMF 5x5 (1)	1.8105	0.14029
WCBiMF 5x5 (2)	1.7569	0.13474
WCBiMF 5x5 (3)	1.7541	0.13307
WCBiMF 7x7 (1)	1.6981	0.12678
WCBiMF 7x7 (2)	1.6945	0.12272
WCBiMF 7x7 (3)	1.6998	0.12195

Figure 8.4: Result from median and WCBiMF filtering of a gap filled disparity map from NCC with 3x3 support window size.

a b c

Figure 8.5: Moebius stereo pair and ground truth. a. Left b. Right c. Ground truth

a b c

Figure 8.6: Dolls stereo pair and ground truth. a. Left b. Right c. Ground truth

Bibliography

[1] Analyzing Octopus Movements Using Three- Dimensional Reconstructions. jn.physiology.org/content/98/2/1775. Accessed: 2015-04-24.

[2] ATI Radeon HD 2900 XT Video Card Review. http://www.legitreviews.com/ati-radeon-hd-2900-xt-video-card-review_503/3. Accessed: 2015-04-01.

[3] Correlation Based Similarity Measures - Summary. https://siddhantahuja.wordpress.com/tag/ncc/. Accessed: 2015-04-16.

[4] GPU- Gems. https://developer.nvidia.com/gpugems/GPUGems/12. Accessed: 2014-12-05.

[5] MATH32031: Coding Theory Part 2: Hamming Distance. http://www.maths.manchester.ac.uk/~pas/code/notes/part2.pdf. Accessed: 2014-01-23.

[6] Red ball 1. http://national-paranormal-society.org/wp-content/uploads/2014/08/red-ball1.jpg. Accessed: 2015-04-01.

[7] Stereo Evaluation. www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo. Accessed: 2015-04-25.

[8] Tsukuba Ground Truth. http://vision.middlebury.edu/stereo/submit/tsukuba/groundtruth.html. Accessed: 2014-02-03.

[9] Wheatstone's Stereoscope. http://www.labgrab.com/timeline/event/charles-wheatstone-shows-one-first-3d-anaglyphs. Accessed: 2015-04-07.

[10] Ery Arias-Castro and David L. Donoho. Does median filtering truly preseve edges better than linear filtering? 2009.

[11] Mike Bailey. Opengl compute shaders. 2013.

[12] J B Calvert. Stereopsis. 2000.

[13] Nico Cornelis and Luc Van Gool. Real-time connectivity cons trained depth map computation using programmable graphics hardware. *IEEE Conference on Com puter Vision and Pattern Recognition. Volume 1.*, 2005.

[14] Angel E. and Shreiner D. *Interactive Computer Graphics, Sixth Edition*. Pearson, 2012.

[15] C J Erkelens and H Collewijn. Eye movements and stereopsis during dichoptic viewing of moving random-dot stereograms. 1985.

[16] Andreas Koschan et al. Color stereo vision using hierarchical block matching and active color illumination. *Proc. 13th Int. Conf. on Pattern Recognition ICPR´96, Vienna, Austria*, 1996.

[17] Ayman Zureiki et al. Stereo matching and graph cuts. 2008.

[18] Dave Shreiner et al. *OpenGL Programming Guide 8th Edition*. Addison-Wesley, 2013.

[19] David Gallup et al. Real-time plane-sweeping stereo with multiple sweeping directions. *IEEE Compute r Vision and Pattern Recogni- tion*, 2007.

[20] Jae Chul Kim et al. A dense stereo matching using two-pass dynamic programming with generalized ground control points. 2005.

[21] Jian Sun et al. Stereo matching using belief propagation. 2002.

[22] Jochen Schmidt et al. Dense disparity maps in real-time with an application to augmented reality. 2002.

[23] Marcus Mueller et al. Adaptive cross-trilateral depth map filtering. 2010.

[24] Qi Zhang et al. 100+ times faster weighted median filter (wmf). 2014.

[25] Robert Haralick et al. Image analysis using mathematical morphology. 1987.

[26] Ruigang Yang et al. Real-time consensus-ba sed scene reconstruction using commodity graphics hardware. *Pacific Graphics 2002*, 2002.

[27] Seongyun Cho et al. Occlusion removal and filling in disparity map for real time multiple view synthesis. 2012.

[28] Sylvain Paris et al. Bilateral filtering: Theory and applications. 2008.

[29] Ziyang Ma et al. Constant time weighted median filtering for stereo matching and beyond. 2013.

[30] Hajer Fradi and Jean-Luc Dugelay. Improved depth map estimation in stereo vision. 2011.

[31] Hieiko Hirchmüller, Maximilian Buder, and Ines Ernst. Memory efficient semi-global matching. 2012.

[32] Heiko Hirschmuller. Semi- global matching - motivation, developments and applications. 2011.

[33] Heiko Hirschmüller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. 2008.

[34] Wenxian Hong. A study of fast, robust stereo-mapping algorithms. 2010.

[35] Karl Jonsson. Shader-based stereo matching with local algorithms. 2003.

[36] Chris McClanahan. History and evolution of gpu architecture. 2011.

[37] K Mühlmann, D Maier, J Hesser, and R Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. 2001.

[38] Jonas Nolte and Magnus Wetterberg. Augmented reality in surveillance systems - a concept study at axis communications ab. 2013.

[39] M Pharr and Randima Fernando. Gpu gems 2. 2005.

[40] D R Proffitt and C Caudek. Depth perception and the perception of events. 2003.

[41] Mikhail Sizintsev, Sujit Kuthirummal, Supun Samarasekera, Rakesh Kumar, Harpreet S. Sawhney, and Ali Chaudhry. Gpu accelerated realtime stereo for augmented reality. 2010.

[42] Richard Szeliski, editor. *Computer Vision - Algorithms and Applications*. Springer, 2011.

[43] Steven Tanimoto and Theodosios Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing, vol. 4, pp. 104-119*, 1975.

[44] M Tornow, M Grasshoff, N Nguyen, A Al-Hamadi, and B Michaelis. Fast computation of dense and reliable depth maps from stereo images. 2012.

[45] Alex Woodie. The modern gpu: A graphic history. 2013.

LUNDS UNIVERSITET
Lunds Tekniska Högskola

MAY 2015

GPU Accelerated Stereo Correspondence

by VILHELM STURÉN AND SEBASTIAN HINDEFELT

Motion detection in video surveillance is easily fooled by animals or shadows, which leads to expensive false alarms. Using a stereo camera for depth perception makes it possible to estimate the size of objects detected in a scene and reduce the number of false alarms drastically.

Artificial Depth Perception

Human vision is a complex system that helps us navigate our surroundings. While most of our vision works well with one eye closed, actual depth perception is completely lost. The reason for this is that we use the relative disparity of objects between the eyes to get information about depth in a scene. This effect can be observed by looking at objects and alternating between the left and right eye, close objects seem to move more between the eyes than objects far away. It is possible to mimic this with two parallel camera sensors that look in the same direction, similar to our eyes, usually called a stereo camera. Because of the vast possibilities and the wide application field, finding depth from cameras is a very active research subject and a huge amount of methods have already been researched. We applied a known method as a base but with completely new modifications involving performance optimization and quality enhancement. Additionally the goal of using a GPU, normally used for computer graphics rendering, is to accelerate the computations, which influenced the choice of methods.

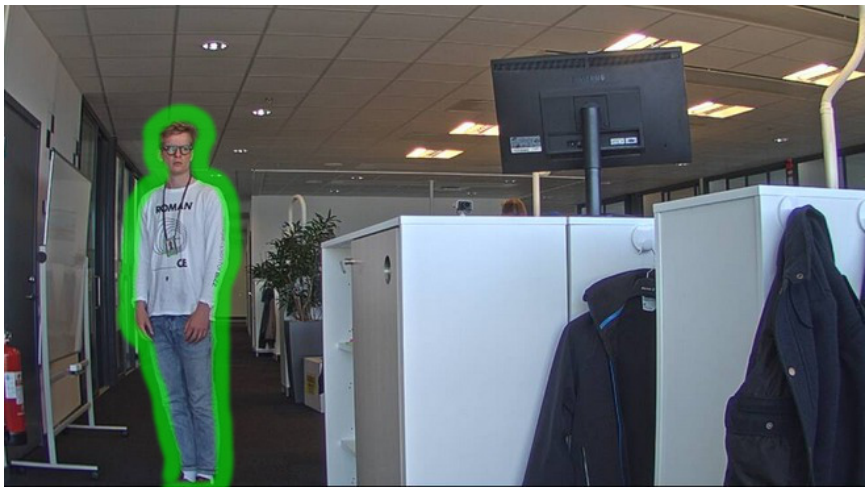
Stereo Correspondence

To find corresponding pixels in a stereo image pair, initially square areas around pixels are compared at the same location in both images. The location in one image is then moved horizontally towards the other image until a match is found. The distance moved corresponds to the depth.

Usually the entire area surrounding the pixel is used to find corresponding pixels but in this study a sparse window is used around the pixel instead. This increased the performance by 35% with minimal quality penalty as compared to the full square around the pixels. This completely new method makes it more realistic to receive depth information in real-time from an embedded camera system.

Object Tracking with Identification

A well known feature in surveillance camera systems is motion detection. The detected objects can be encapsulated inside an area which gives an operator information about where on the screen the objects in motion are found. This information in combination with depth perception makes it possible to present the real size of a detected object and identify what kind of object it is. Knowing the type of an object can be further useful to filter out false alarms otherwise triggered by motion detection, for example from animals or shadows, which cost a lot of money and are a waste of time for the people involved.



Motion detection aided by depth (concept)

Vilhelm Sturén: *ada09vs1@student.lu.se*
Sebastian Hindefelt: *ada10sh1@student.lu.se*